University of Alberta

A Study in Real-time Collaboration Systems

by

Xiaolin Qiu

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

Department of Computing Science

Edmonton, Alberta
Fall 1996

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-18315-7

Canada

University of Alberta

Library Release Form

Name of Author: Xiaolin Qiu

Title of Thesis: A Study in Real-time Collaboration Systems

Degree: Master of Science

Year this Degree Granted: 1996

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Xiaolin Qiu
#206, 10725 - 83 Avenue
Edmonton, Alberta
Canada, T6E 2E5

Date: June 21, 1996

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **A Study in Real-time Collaboration Systems** submitted by Xiaolin Qiu in partial fulfillment of the requirements for the degree of Master of Science.

Dr. Anup Basu (Supervisor)

Dr. Bruce Bentz (External)

Dr. Janelle Harms (Examiner)

Date: June 19, 1996

To my husband, Dr. Yaoqing Gao
my son, Richard Gao
my parents,Yayuan and Shigang Qiu

Abstract

People collaborate because collaboration changes the process of work for them in desirable ways. The rapid spread of computer and telecommunication technologies has made individual use of computers in the home and office commonplace. Computers are utilized for many routine tasks, much collaborative or cooperative work is computer-based. Computer supported collaborative systems provide easy, interesting, and effective group interactive environments.

In computer supported collaborative systems, the mode of interaction is either asynchronous or synchronous (real-time). This thesis research focuses on the real-time collaboration systems, which are multi-user computer applications that allow physically distant people to work together at the same time. The main goal of this research is to develop a real-time collaboration system for real-time work. The interaction and distribution technologies underlying real-time collaboration systems are addressed. This thesis also describes the design and implementation of a real-time collaboration system.

Acknowledgements

I would like to take this opportunity to express my sincere thanks to those who have helped me throughout my graduate studies.

First of all, I would like to express my deepest gratitude to my supervisor, Dr. Anup Basu, for his invaluable guidance, encouragement, and patience throughout the research work. The work could not have been completed if it was not for his supervision. It was a great pleasure working with and learning from him.

I am grateful to the members of my examining committee, Dr. Bruce Bentz and Dr. Janelle Harms for their careful reading and valuable comments on my thesis, and for serving as my examining committee members. Thanks also go to Dr. Hong Zhang who chaired my defense.

Special appreciation is due to all members of the Computer Vision Group. They are all very friendly people to work with. I thank Ms. Anne J.R. Nield for her proofreading of the thesis. You made my thesis look much better, Anne. In particular, I am grateful to Kevin Wiebe for his useful discussions and his face display programs. He was a good colleague.

Many thanks to my parents and my son. They have given me great support.

Thanks especially to my husband, Yaoqing Gao, whose endless support and encouragement made all of this possible. I am greatly indebted to him!

I would also like to express my appreciation to Dr. Anup Basu and the Computing Science Department for their financial support.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Thesis Organization 3

2 Background **5**

2.1 Classifications 6

2.1.1 Functionality 6

2.1.2 Geographic 7

2.1.3 Time 7

2.2 Examples 7

3 Collaboration Application **11**

3.1 Overview 11

3.2 Fundamental Concepts and Characteristics 12

3.2.1 Characteristics 13

3.2.2 Application Scenario 14

3.3 Requirements for Collaboration Systems 15

4 System Design and Functionality **17**

4.1 Collaboration System Design Issues 17

4.1.1 Interaction technology 18

4.1.2 Distribution Technology 20

4.2 System Functionality and Design Considerations 25

4.2.1 System Functionality 25
4.2.2 Design Considerations 26

5 N-Talk: A Text Communication Service **29**
5.1 Overview 29
5.1.1 Features 30
5.1.2 Interface 32
5.2 Implementation Details 40
5.2.1 Registration 42
5.2.2 Motif User Interface 43
5.2.3 N-Talk Client 43
5.2.4 N-Talk Server 45
5.2.5 Message Transfer Primitives 48
5.2.6 Concurrency Control 48
5.3 Generalizing Design 48

6 Integration of Image Processing and Text/Picture Editing **50**
6.1 Co-Draw: A Collaborative Drawing Tool 50
6.1.1 Overview 50
6.1.2 Implementation Details 51
6.2 Co-Image: An Image Display and Manipulation Tool 53
6.2.1 Functionality and Features 53
6.2.2 Implementation Experiences 56
6.3 Co-Edit: A Collaborative Edit Tool 59
6.3.1 Overview 59
6.3.2 Requirements for Collaborative Writing 60
6.3.3 Distemacs Co-Edit Tool 60

7 Conclusion and Future Work **67**
7.1 Conclusion 67
7.2 Future Work 68

List of Tables

5.1 Communication Primitives for N-Talk 49

6.1 Communication Protocol between Processes 52

6.2 Detailed List of Items 52

List of Figures

4.1 The Centralized Architecture 21
4.2 The Replicated Architecture 22
4.3 The Semi-replicated Architecture 23
4.4 Example of Operations Re-ordering 24
4.5 Collaborative System Architecture 27
4.6 Collaborative System Main Interface 28

5.1 N-Talk Architecture 30
5.2 Create Session Control Flow 31
5.3 N-Talk Control Window 33
5.4 Session Name Prompt Window 36
5.5 Session Selection Window 36
5.6 Invitation Window 37
5.7 Leave Request Window 37
5.8 Username Prompt Window 37
5.9 Quit Question Window 38
5.10 Quit Error Information Window 38
5.11 The Control Window Shows All Sessions 39
5.12 AllUser Faces 39
5.13 One User Face 40
5.14 Talk Text Windows 41
5.15 Data Structures 47

6.1 Co-Draw Windows 54
6.2 The Original Image 57
6.3 The Updated Images after cut and paste 57
6.4 The corrected image 58
6.5 The updated images after gamma correcting the image 58
6.6 Increase in internationally co-authored articles 59
6.7 Two People Using Distemacs Editor simultaneously 62
6.8 Distemacs Control Window 63
6.9 Typical structure of a single-user editor 64
6.10 Structure of an Editor built using the DistEdit toolkit 65

Chapter 1

Introduction

1.1 Motivation

One of the major reasons for the recent popularity of Computer Supported Cooperative Work(CSCW) is the opportunity it provides for easy, interesting, and effective group interaction environments. People collaborate because collaboration changes the process of research for them in desirable ways. They believe that working with others improves the quality of the research product because of the synthesis of ideas it affords, the feedback they receive from each other, and the new skills they learn. Through collaboration people can tackle problems that they are incapable of working on alone because of the limitations of their resources, knowledge, skills, or time.

Effective business practice demands contributions from a wide variety of people, with diverse backgrounds. The manufacturing industry, commerce, research institutes, and governments all require coordinated teams to research, develop, and communicate various kinds of information. Traditionally, members of the teams were located close to each other and discussed projects face-to-face in the same room.

Adapting to recent trends, establishing plants, sales branches, and research laboratories outside an organization's home country has become popular for carrying out research, manufacturing products, marketing, and providing services in the most advantageous locations. It is becoming increasingly common for manufacturing design

teams to be composed of members belonging to the same organization, but located in different places.

With research constantly widening the scientific horizon beyond the scope of the most capable single mind, and with increasingly complex research topics, scientists need to cooperate to deal with multi-faceted problems. Based on the shared understanding of those problems, it is more possible to develop solutions.

The rapid spread of computer and telecommunication technologies has made individual use of computers in the home or office commonplace. Computers are utilized for many routine tasks, and a lot of collaborative or cooperative work is computer-based. To respond to the increasing research and development activity associated with the augmentation of group work by computers, the name CSCW (Computer Supported Cooperative Work) emerged. Irene Greif (at the Massachusetts Institute of Technology) and Paul Cashman (of Digital Equipment Corporation) organized a workshop in 1984, where they first used the term Computer Supported Cooperative Work (CSCW). They shared concerns about supporting teams working together with computer systems. Throughout the 1980s, a number of technological developments occurred which established the preconditions for the emergence and rapid growth of CSCW.

Collaboration is described by Goodman and Abel as a process involving people sharing information of some form and thus effecting changes in the thinking and actions of most people [18]. Collaborative systems allow group members to discuss a problem and actually create something together. Such systems are one of the important applications of multi-user multimedia systems. Considerable recent research and developmental efforts have been directed towards multi-user multimedia systems. Computer supported cooperative interaction, incorporating information exchange and multimedia communication, will revolutionize collaboration in scientific and engineering settings. (Collaboration and cooperation are equated in concept in this thesis.)

Real-time computer supported collaboration allows a group of users, who are either gathered in an electronic meeting room, or physically dispersed, to interact

synchronously through their workstations or terminals. Collaborative computing already affects us. Real-time computer collaborative applications have been a major focus of groupware development efforts, often motivated by CSCW.

Cooperative work means multiple individuals working together in a planned way in the same production process, or in different but connected production processes.

Of course, there are many potential applications for real-time multimedia collaboration. For example, in CIMS (Computer Integrated Manufacturing Systems), collaboration plays an important role. Computer Integrated Manufacturing (CIM) uses computer based systems to integrate, monitor, and control all the functions of a manufacturing oganization. This includes all activities from design and production, handling and quality control, to distribution, marketing, and financial control.

Currently we face a significant challenge in making a successful transition to a knowledge-based economy and learning society. Propelled by competition at a global level, organizations are attempting to improve efficiency and effectiveness by integrating the processes of learning, work, and decision making [10]. Empowering people to respond to these challenges requires the creation of infrastructures for learning that extend beyond the classrooms of traditional educational institutions to the desk tops and work benches. Tele-learning is a way to support people in learning work-related knowledge outside the traditional classroom. Tele-learning systems which employ computer-mediated telecommunications are collaborative education systems.

The objective of my thesis is to develop a collaborative multimedia environment that would be a first step towards developing sophisticated applications for Tele-learning and Integrated Manufacturing Systems.

1.2 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 gives a brief overview of related work in the CSCW area. First, the definition of CSCW is discussed. Second, the classifications of the CSCW systems are given. Then, some example systems are

presented.

Chapter 3 introduces collaboration applications. In this chapter, fundamental concepts for collaboration applications are given, and a set of characteristics for collaboration applications is presented. Some general design requirements for collaborative systems are described.

In Chapter 4, the design issues of collaboration systems are discussed, and collaboration awareness is addressed. Three architectures of collaboration systems are presented in detail, and there is a discussion of concurrency control is discussed. Then, the functionality of our real-time collaboration system and design considerations are described.

The text communication service, N-Talk, is discussed in Chapter 5. The features of this service are described. A detailed description of the interface is presented. Co-Draw, Co-Image, and Co-Edit application tools are presented in Chapter 6, and their features and implementation details are discussed. Finally Chapter 7 summarizes the thesis, describing possible enhancements and future developments.

Chapter 2

Background

The rapid evolution of information and the new potential for communication between people have both been of great importance to the success of most organizations. Key aspects are the increased availability of computer networks, and the trend towards team work. One of the main emphases is on computer support for team work. Activities in that domain are known by the terms, *groupware* or computer-supported cooperative work (CSCW). Since CSCW was first used by Irene Grief and Paul Cashman in 1984, much research has been done in this area and numerous systems have been developed.

What is CSCW? A consensus on a definition for CSCW does not yet exist—although there is no shortage of contenders. Dix *et al.* [12] gave one definition for CSCW. They explain that CSCW "is about groups of users—how to design systems to support their work as a group and how to understand the effect of technology on their work patterns". According to this definition, collaboration is an important characteristic. Wilson [43] defines CSCW as "CSCW—a generic term which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques." Another short definition for CSCW is "the study of how people work together using computer technology". These definitions all include the concept, "work as a group". The term, *groupware* is often used in articles in the computing press to represent

the range of existing multi-user networked products. Like CSCW, *groupware* is also difficult to define. No one definition satisfies everyone involved. Robert Johansen gave a definition for *groupware* in his book of the same name [23] as "a generic term for specialized computer aids that are designed for the use of collaborative work groups. Typically, these groups are small project-oriented teams that have important tasks and tight deadlines. *Groupware* can involve software, hardware, services, and/or group process support". Ellis [15] defines *groupware* as "computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment." Normally, *groupware* is used to specifically denote the technology that people use to work together, whereas "CSCW" refers to the field that studies the use of that technology. The software developed for collaborative environments is called *groupware*. In this thesis, we use both terms.

2.1 Classifications

In this section, we classify the CSCW systems by functionality, geographical space, and time.

2.1.1 Functionality

Audio/Video conferencing systems allow two or more users to communicate with live video images, audio signal, or both.

Co-authoring and shared writing tools are multi-user editors. One or more users can be editing the same document.

Meeting room systems provide support for face-to-face meeting in specially designed meeting rooms with a large screen and a number of workstations/PCs. Normally two or more types of applications are used—such as audio/video conference and drawing tools.

Sketch pads and whiteboard systems provide the participants with a "shared whiteboard" where everybody can draw or write in a shared space.

Conversation applications are being developed to help individuals converse with each other.

Message systems support the asynchronous exchange of textual messages between groups of users.

2.1.2 Geographic

One of the important features of CSCW systems is the geographic distribution of the users. Local collaboration occurs in the face-to-face environment. Remote means that users are in different locations.

2.1.3 Time

An important characteristic of CSCW is the mode of interaction, which is often classified as being either synchronous or asynchronous.

For synchronous mode, all the information is exchanged between participants in real-time. Two or more users share the same object in different time in an asynchronous system. The asynchronous collaboration is non-real-time interaction.

2.2 Examples

There are many systems which have been developed in the CSCW area. The following briefly describe several applications.

We classify the synchronous mode as local or remote according to geographical distribution. A good example for synchronous-local collaborative systems is a computer-supported meeting environment, in which each attendee has a computer and had a large public screen at the front of the room. Some of these systems are:

- **Colab** developed by Xerox PARC, that provides real-time services. Colab allows two to six people to collaborate using personal computers interconnected over a LAN [25][37][38]. There are several tools in the **Colab** system. **Cog-**

noter is a tool for brainstorming, organizing and evaluating ideas. **Argnoter** was developed for presenting and evaluating proposals, and **BoardNoter** is a less well developed tool for freestyle sketching.

- **Capture Lab** developed at Electronic Data Systems. The **Capture Lab** system has many similar features to the **Colab** system. The **Capture Lab** implements a shared hardware approach to management of computer resources [35].

In a synchronous-remote collaborative system, the participants are located in geographically different places. This is a distributed shared workspace which provides a multimedia environment where each member can simultaneously see the same thing. Individuals in different locations can work together in real-time. A number of these systems have been developed. Examples of such systems include:

- **MMConf**, implemented in the Diamond System [4][9], which provides a shared display of a multimedia document, as well as communication channels for voice and shared pointers.

- **Rapport** system — a multimedia conferencing system developed at AT&T [1]. The Rapport system supports various forms of interaction, from simple telephone-like conversations to multi-party shared-display interaction.

- **MERMAID** [42] provides an environment for widely separated participants to hold real-time conferences by interchanging information through video, voice, and multimedia documents. The computer hardware consists of UNIX workstations equipped with a variety of advanced input and output devices.

- **RTCal**(Real Time Calendar), developed at MIT [4], is a meeting scheduling system.

- The European collaborative projects **MIAC**(Multipoint Interactive Audiovisual Communications) and **MIAS** (Multipoint Interactive Audiovisual System) [8], provide useful conference management facilities.

- **Shastra** [3], developed by Purdue University—an extensible, distributed, and collaborative geometric design and scientific manipulation environment. It consists of a static and dynamic component. The static component, the shastra layer, is a CSCW infrastructure for building scientific CSCW applications. It defines an architectural paradigm that specifies guidelines on how to construct applications which are amenable to inter-operation. Its connection and distribution substrate facilitates inter-application cooperation and distributed problem solving for concurrent engineering, while its communication substrate supports transport of multimedia information. The collaboration substrate supports building synchronous multi-user applications by providing session management and access regulation facilities. In addition to the distribution, communication, and collaboration framework, Shastra provides a powerful numeric, symbolic, and graphics substrate. It enables rapid prototyping and development of collaborative software tools for the creation, manipulation, and visualization of multi-dimensional geometric data. The dynamic component of Shastra is a runtime environment that exploits the benefits of the architectural philosophy and provides runtime support for conference applications [2].

- **VideoDraw** and **VideoWhiteboard** [39][40] use video-based technology to create distributed shared drawing surfaces which support the use of hand and body gestures in the collaborative drawing process. **VideoDraw** is a two-person desktop system. **VideoWhiteboard** is a video-based prototype tool that provides a large area of shared drawing space between two geographically separated groups.

- **TeamWorkStation** — designed to provide for small geographically separated work groups(2-4 members). **TeamWorkStation** provides users with a shared screen as the open shared workspace, and live video and audio communication links for face to face conversation [21][22].

In asynchronous mode, people interact over an extended period of time such as in

postal correspondence. E-mail was the first widely available form of a CSCW system.

It is clear that the asynchronous mode is the most successful. However, in this thesis we focus on synchronous collaboration.

Chapter 3

Collaboration Application

3.1 Overview

Collaboration relies on a shared space. It may be a room, a blackboard/whiteboard, or a shared on-line space. Shared space serves as a touchstone for the act of collaboration, and it is essential as a medium to manage the ambiguity inherent in human interaction [17]. In effect, these shared spaces are the collaborative tools that provide a context in which the whole of the relationship is greater than the sum of the individual participants' expertise.

Computers significantly affect how we interact and collaborate. Traditionally, computers have allowed asynchronous group work through shared file systems, explicit file transfer, and electronic message exchange. Real-time computer supported collaboration systems allow physically distant people to work together in a shared space at the same time.

Collaboration applications are designed to be used by a group. They contain functionality for handling issues that emerge when such a group interacts through networked computers. They fall into a relatively new research field called Computer-Supported Cooperative Work(CSCW). CSCW has emerged as an identifiable research area which focuses on the role of the computer in group work. The goal, from a computer science perspective, is to create hardware devices and software programs

that allow end-users to share information in order to help them work together more efficiently and effectively.

3.2 Fundamental Concepts and Characteristics

Before we discuss the collaborative applications in more detail, we will discuss some fundamental concepts first. These concepts are:

Real-time: The delay between an input and its resulting output must have an upper bound commensurate with human interactions. Ideally the delay should not be noticeable. The sequencing relationships within and between input and output data streams must be preserved [28].

Shared context: A shared context is a set of objects where the objects and the actions performed on the objects are visible to a set of users.

Collaboration: People sharing information of some form and thus effecting changes in the thinking and actions of the people involved in the process[18]. Collaboration requires individuals to work together in order to achieve a single common goal.

Group window: A group window is a collection of windows whose instances appear on different display surfaces. The instances are connected. For example, drawing a circle in one instance makes a circle appear in the other instances.

View: A view is a visual, or multimedia representation of some portion of a shared context. Different views may contain the same information but differ in their presentation (for instance, an array of numbers can be presented as a table or as a graph), or they can use the same presentation but refer to a different portion of the shared context.

Session: A period of time when two or more members of a group are working together synchronously.

3.2.1 Characteristics

In this section we discuss a set of characteristics for collaboration applications.

Interaction Characteristics

The basic characteristics of interactions can be identified as:

- synchronous interaction

  Synchronous interaction implies interactions of all group members at the same time (e.g. in a video conference).

- asynchronous interaction

  Asynchronous interaction allows participants to interact within a certain time frame (e.g. electronic mail).

Most collaboration systems support only one of these interaction modes.

Communication Characteristics

The basic communication characteristics include:

- open/close a communication channel for a group of participants.
- administration of a session.
- reliable sending and receiving of data from one or many participants.
- consistency.

Media Types

The media types of a collaborative application can be one of the following types or their combination:

- *text*, which includes different fonts, sizes, styles, and formatting information. Talk and co-author editing systems support this kind of media type.

- *image*, which includes grey-scale or color information.
- *line graphics*, including grey-scale or color information for line and filling patterns, such as shared draw systems.
- *spreadsheet*, including calculation capabilities, as well as graphical data presentation, which is sensible to changes in the underlying data in the spreadsheet.
- *audio*, which provides the sound facility during a session.
- *video*, including live video images which are used in video conference systems.

3.2.2 Application Scenario

We would like to use an example of a collaborative project in an academic/research environment to demonstrate how a collaboration application works. The following example scenario illustrates collaboration between users. It is an informal, interactive meeting to start a new research project.

> In this scenario, the potential members of a project team first come together. They are located in different places. The goal which they want to achieve is to create a first draft of the project outline. To support this activity, a talk session may be created to allow the participants to exchange ideas, opinions, and suggestions with each other. They may want to draw some informal structure figures. They may want to write a project outline by using a co-authoring editing tool. In terms of the characteristics involved, the application will be synchronous, remote, and multi-way. The main media types required will be text and line graphics. These two media types can be transmitted quickly since they have low bandwidth requirements.

The above scenario is only one example of many possible collaboration applications. Some of others are:

- group decision support environment.
- multi-user cooperative design.
- screen/window sharing facility.
- shared calendar system.
- audio/video conferences.

3.3 Requirements for Collaboration Systems

Some general design requirements for collaborative systems are [33]:

- To support different registration methods: for some sessions, anyone is allowed to join. For others only a select group can participate.
- To support latecomers to the session: a user can join an ongoing session. The system should support the current session state to the newcomer.
- Participant termination: a participant can leave a session at any time.
- Integration of several media types: multimedia collaborative application systems provide more effective communication between the participants.
- Provide the session management: The system should oversee all the session management, which includes activities such as participant registration, session information, communication and so on.
- To provide concurrency control: To prevent the conflicting actions on the same item or point.
- Group interfaces: group interfaces differ from single-user interfaces in that they depict group activity, and are controlled by multiple users rather than a single user.

- Collaboration awareness: the system should support collaboration awareness. When a user performs an action, all other users in the same session can update their output.

- Error independence: a local user interface error should not affect other users and the program.

- User interface independence: the system should allow the users to see the different screen images. For example, it allows the users to scroll to different parts of a document.

Chapter 4

System Design and Functionality

4.1 Collaboration System Design Issues

In the previous chapter, the various requirements for collaborative application systems in general were examined. In this chapter, we will discuss the design issues of collaboration systems. Why is real-time collaborative system development difficult? For a start, collaborative system developers face all the problems experienced by single-user application developers. Besides all the normal problems of building single-user applications, the collaborative system developers must be concerned with technical issues such as synchronization, concurrency, communication, registration, and more. Several authors have attempted to organize the technical issues that serve as a foundation for collaboration applications. According to these requirements, which we examined in the previous chapter, we can categorize the technical issues for real-time synchronous collaboration into two disciplines:

- Interaction technology
- Distribution technology

4.1.1 Interaction technology

This category includes all technologies that are involved in human-human and human-computer interaction. Typical examples are the window system, discrete (graphics, text) and continuous (audio, video) media, and window coupling.

User Interface System

The user interface manifests itself in most computer systems as an identifiable component dedicated to serving the needs of user interaction.

The Sequential View: Single User and Single Application. Initial interface systems focused on connecting a single user to a single application, and concentrated on the interaction between the user undertaking a task and the application which supported this task.

The Asynchronous View: Single User and Multiple Applications. As computers grew in power a user could interact with a range of different applications. The generation of user interface focus was on the support of a range of applications.

Multiple User Interfaces. In the CSCW area, people requires a wide range of cooperative applications with multi-user interfaces. There are two approaches for the development of such interfaces. The first approach is *collaboration transparent.* This approach adapts existing single user applications to collaborative applications by merely replicating the display portion of existing single user interfaces without altering the interface. The second approach is *collaboration aware.* The collaboration aware solution provides facilities to explicitly manage the interface between different users.

Collaboration Awareness

Computer supported cooperative work requires the construction of applications which support interaction by multiple users. These applications exploit multi-user interfaces to promote their cooperative use by a community of users. Users may be located in distributed places and the associated interfaces run on a number of workstations.

Collaborative applications should provide users with an awareness of the activities of others in order to support and encourage cooperation to take place. In these applications, the purpose of a multi-user interface is to establish and maintain a common context. This context allows the activities of one user to be reflected on other users' screens, and is supported by sharing application information. This sharing is the principle means of promoting cooperation, and the real-time presentation and manipulation of shared information is the main function of cooperative, multi-user interfaces [36].

Interaction coupling (or interface coupling) is an important issue in collaboration. Different forms of collaborative work require varying levels of awareness between users. The greater the level of awareness, the closer the interaction coupling. There are three different degrees of interaction coupling which correspond to different levels of awareness between users [36][41].

- *Tight coupling:* Each user is presented with the same display of the same subset of a common information space. Every action performed by any user is immediately observable by the other users. This is strict WYSIWIS (What-You-See-Is-What-I-See – pronounced "whizzy whiz") [37][38].

- *Medium coupling:* Each user may utilize a different presentation method to display the same subset of a common information space. For example, different users may simultaneously interact with tabular or graphical display of the same data.

- *Loose coupling:* Each user maybe presented with a different part of the shared information space. For example, if a group of authors use the Co-Edit tool to

edit a shared paper, they may edit different sections of the paper.

Strict WYSIWIS is often too restrictive and much work is needed to support more customizable interaction coupling [6][11][32].

4.1.2 Distribution Technology

The architecture of collaborative systems influences their consistency and synchronization. There are two main architectures for distributed collaborative systems.

Centralized Architectures

In a centralized architecture (Figure 4.1), a single copy of the application exists in the central server. The central server program handles all user input events and distributes output to all session sites. Local workstations act as graphical terminals and window servers. The primary advantage of the centralized architecture is its simplicity; it is easy to implement, keep it consistent, and add/remove displays. However, centralized architectures demonstrate poor performance as the workloads for the server program are too heavy. They need high bandwidth since the server needs to broadcast both input and output to all the users. In addition, the response time is long, and they are vulnerable to failure (either machine or network). It is impossible to support different views of the shared information within these systems.

Replicated Architectures

A fully replicated architecture is used to achieve minimized response time. This architecture maintains exact copies or replicas of the application on each workstation. Servers at each site distribute input to each replica. Some control mechanism is necessary to ensure that all users in the same session see the same output, if they wish. Replicated architectures provide good performance. They require low bandwidth because only input must be distributed among the users. Since this approach involves local management of the display, it is easy to support the different views.

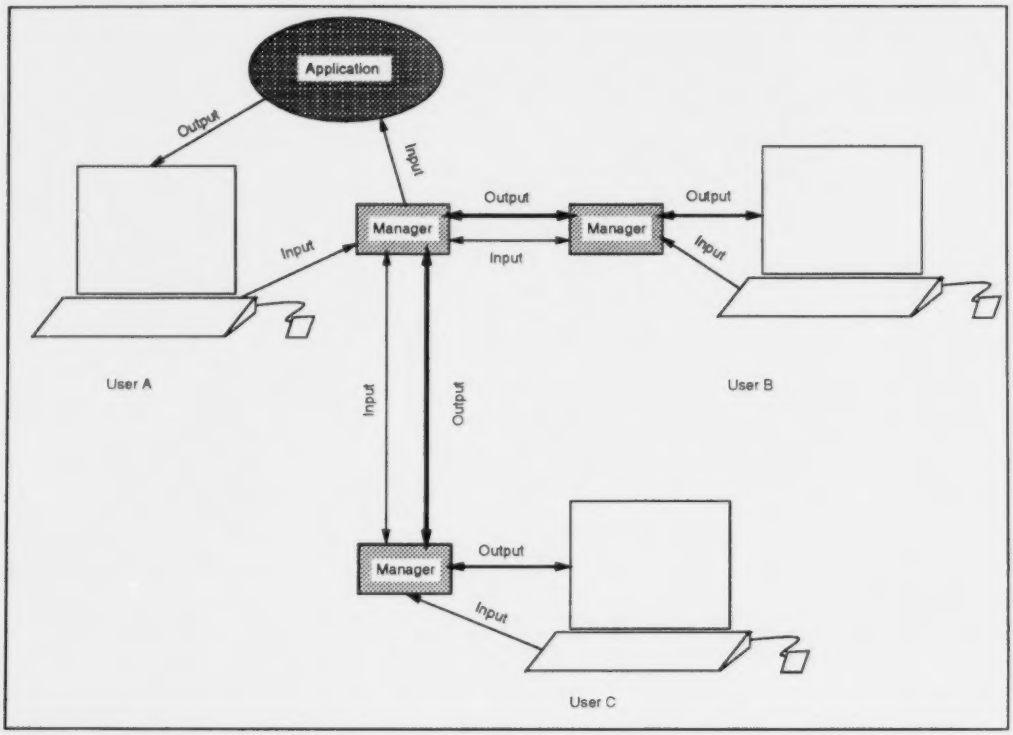

Figure 4.1: The Centralized Architecture

The major difficulties with replicated architectures concern synchronization and data consistency. Users can perform actions simultaneously which are executed locally before being broadcast to other machines. If these actions conflict — for example, one user deletes a selected object in a WYSIWIS group drawing program at the same time as a second user changes the selection to a different object — inconsistent interface can result due to events arriving in a different order at each machine. To prevent such conflict requires complex synchronization algorithms.

The replicated architecture is shown in Figure 4.2.

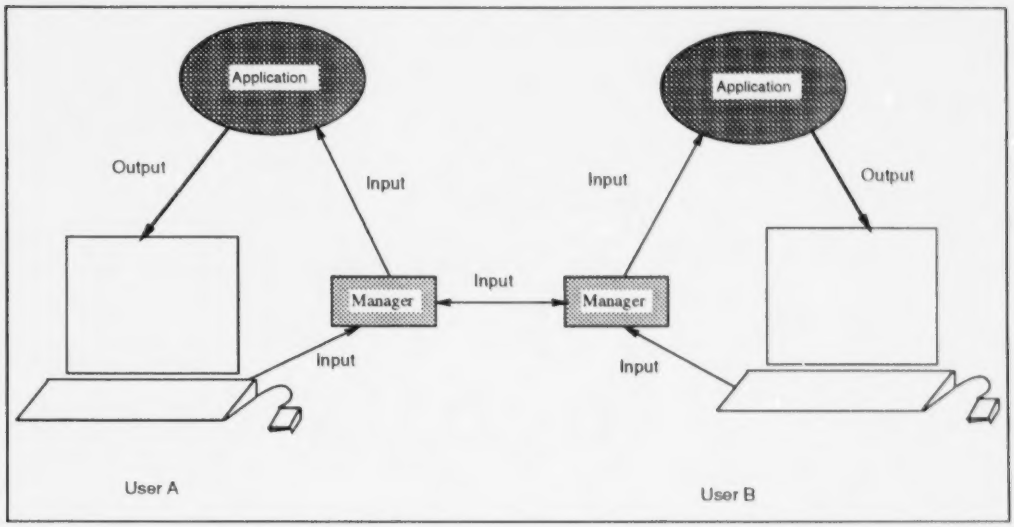

Figure 4.2: The Replicated Architecture

Semi-replicated Architectures

Both centralized and replicated architectures offer benefits and limitations. The semi-replicated architecture is proposed to overcome the limitations. This architecture provides a good solution as it manages to keep communication, consistency, and synchronization cost at a low level. Figure 4.3 shows the semi-replicated architecture. This is a client/server model. The server workstation is responsible for registration, synchronization, concurrency, etc. The application program runs on client machines. Each user works on a client machine which communicates with the server to cooperate with other users. Two goals form the key to this architecture:

- to put as much useful computation as possible on the client machines;
- to keep the network traffic to a minimum so that the system is scalable.

The bottleneck of a client/server model is that adding new clients increases the load on the server and the network. Since this architecture puts all the computation on the client side, the output is not transferred. This keeps network traffic low since network communication is low, so it is scalable.

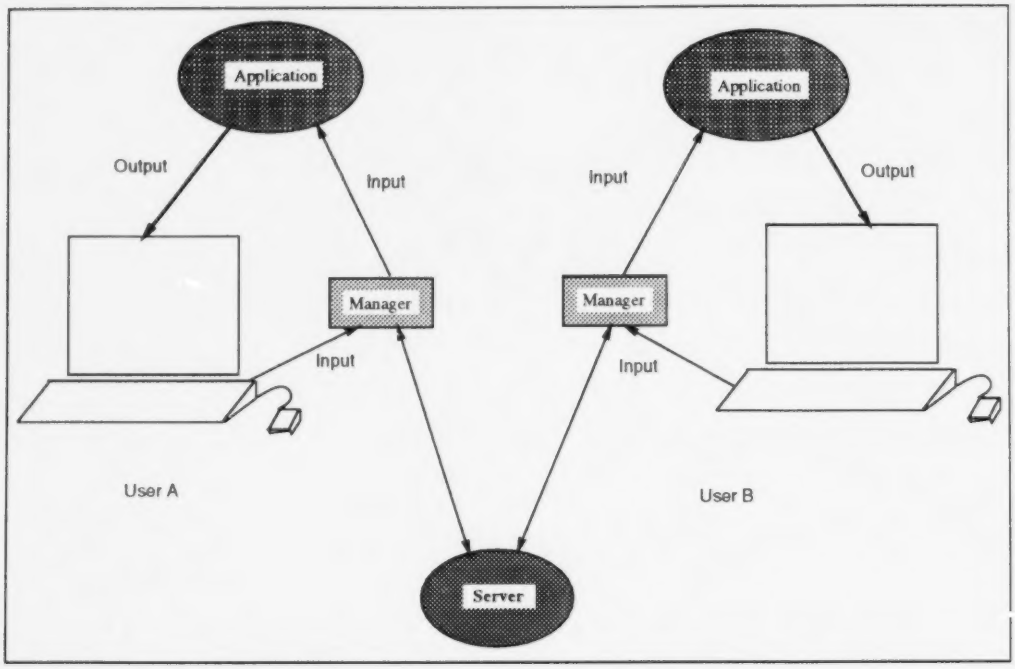

Figure 4.3: The Semi-replicated Architecture

Concurrency Control

Concurrency control is the activity of coordinating the potentially interfering actions of processes that operate in parallel [19]. Figure 4.4 shows a simple example of two sites losing integrity. In site A, Operation 1 is followed by Operation 2—but Operation 2 is followed by Operation 1 at site B. The different order of operations may result in both sites being inconsistent and out of step. For example, one participant added a word to a sentence at the same time as another participant removed this sentence. After the two operations, the displays are different at the two sites. There are two types of concurrency control approaches: pessimistic and optimistic. The optimistic approach determines how events can be received. This method allows events to be received out of order. Inconsistencies must eventually be detected and repaired. The pessimistic policy ensures that events can only be received in order, thus guaranteeing consistency.

Locking (or floor control) is a common approach to concurrency control. Typically, a user will request a lock to an object. If no one else holds the lock, the request is approved and the user gains the lock. The user can then manipulate the object. If someone else is holding the lock, the request of the user is denied.

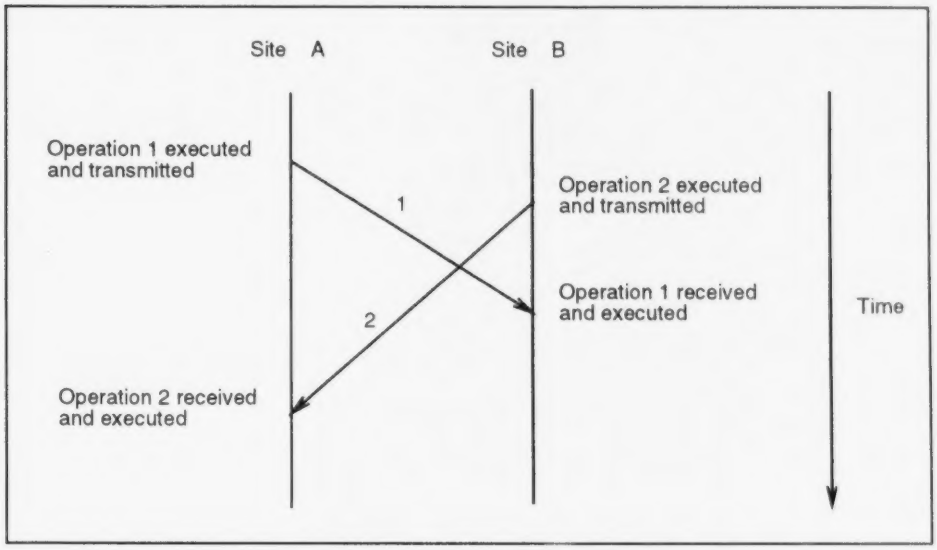

Figure 4.4: Example of Operations Re-ordering

The pessimistic locking approach forces a user to wait until a lock request is answered and creates a locking message overhead. If delays are barely noticeable, this may not matter. However, when the network or processors suffer a visible delay, the approach may translate into a poor interface for the user. Because of its simplicity and ease of implementation, a pessimistic locking policy is chosen by many applications.

The optimistic approach is based on the assumption that conflicting events are rarely received out of order. Optimistic locking assumes that the user will frequently be granted his/her lock requests. After a user requests the lock, he/she starts manipulating the object before he/she knows if he/she really has the lock. If the lock is granted, the work continues as normal. If it is denied, the object must be returned to its original state, which is difficult to do. It is not clear what to do when locks are denied.

The other issue in concurrency control is granularity size. Different grain sizes give a very different feel to collaborative systems. For example, a fine grain size means there will be many lock requests, and the lock message overhead will be high. Coarse granularity implies fewer lock requests, but less opportunity for concurrency as locks will be denied more frequently. The choice is a balance between locking overhead and the amount of concurrency desired.

4.2 System Functionality and Design Considerations

4.2.1 System Functionality

The system functionality of our collaborative system falls into the following main categories: communication service, group drawing, co-authoring writing, group image processing.

- communication service.

  The N-Talk service allows the users to talk with each other. This communication service provides a facility to users to exchange ideas, opinions and suggestions as they use other tools. The users can use it to "chat".

- group drawing.

  The Co-Draw program provides a tool to enable group users to do a drawing.

- group interactive image display and manipulation.

  The Co-Image program supports interactive image display and manipulation functions, such as cut, paste, copy, chop, edge detection, etc. between group users.

- co-author writing.

  The co-authoring writing program supports group writing.

4.2.2 Design Considerations

Computer Requirements

The system was developed for UNIX, TCP/IP environments. We use sockets to connect the users' processes. Sockets allow programmers to open reliable, connection-oriented communication between two running processes, creating a virtual point-point communication. Communication normally flows to many user processes with real-time constraints. We use the client/server model to implement each of our tools. The client/server applications can be divided into single-server applications and multi-server applications. The system is implemented by using the multi-server model, because we want to reduce the workload of the servers and to achieve a small response time. Each tool has its own server.

N-Talk, Co-Draw, and Co-Image are implemented by using the semi-replicated architecture. There are two main reasons for choosing this architecture. First, from the communication side, there is a copy of the application running at every site in a session; therefore, we only need to send the input between users. This reduces the network bandwidth and the response time. For example, in Co-Image, suppose we want to cut a part of an image—we just need to send the start point (x, y), the height, and width of the area which we want to cut, to the other participants. If we use centralized architecture, we would need to send the updated image, which needs high bandwidth and longer transfer time.

The second reason for choosing semi-replicated architecture is for concurrency control. On the surface, the simplest way of implementing concurrency control is through a centralized architecture. The centralized approach uses a single application program to control all input and output to the distributed application. As we have already mentioned, the centralized architecture needs high bandwidth and has a long response time. In the semi-replicated architecture, there is a centralized server to broadcast the input and control the order. Given that we use the semi-replicated architecture, we do not use any special concurrency control policy to handle the

concurrency problem in Co-Draw. The system structure is shown in Figure 4.5.

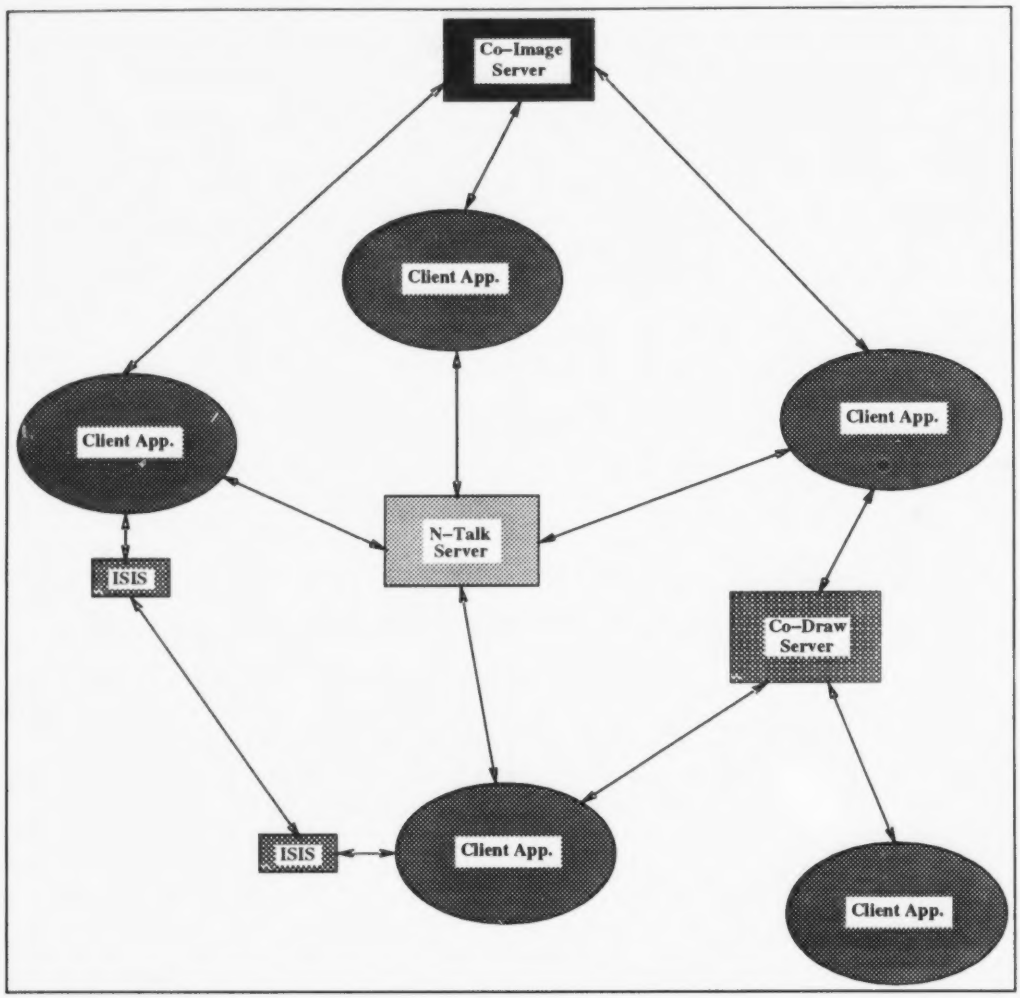

Figure 4.5: Collaborative System Architecture

Programming environment

- Programming Languages

  Our tools are programmed in C programming language.

- Graphical User Interface Environment

  We chose Motif to implement the graphical user interface. Motif is one of the

major interface standards in the UNIX world. It is based on the X Window System which is a popular graphical environment. Motif fits in reasonably well with X standards and with the use of window managers and resource files [24].

Figure 4.6 shows the first interface. Depending on what the user wants to do, he/she pushes an appropriate button to connect with a specific server. A user does not need to have knowledge about the server. For example, if the user wants to utilize the drawing program, he/she clicks the Co-Draw button to connect with the drawing server.

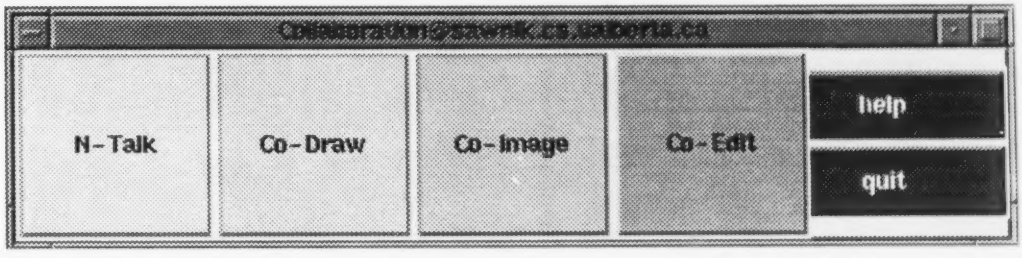

Figure 4.6: Collaborative System Main Interface

Chapter 5

N-Talk: A Text Communication Service

5.1 Overview

There are *talk* programs that are very interactive but there usually are restricted to two users. In this chapter we present a text communication talk service. We call it N-Talk since it supports synchronous n-way textual conversations. The reasons we developed N-Talk are:

- It can be utilized by users who do not have multimedia communication facilities on their desktop.
- It provides a communication method to users who are using other collaborative tools in our system to produce joint work through real-time collaborative sessions.

The architecture of N-Talk is shown in Figure 5.1.

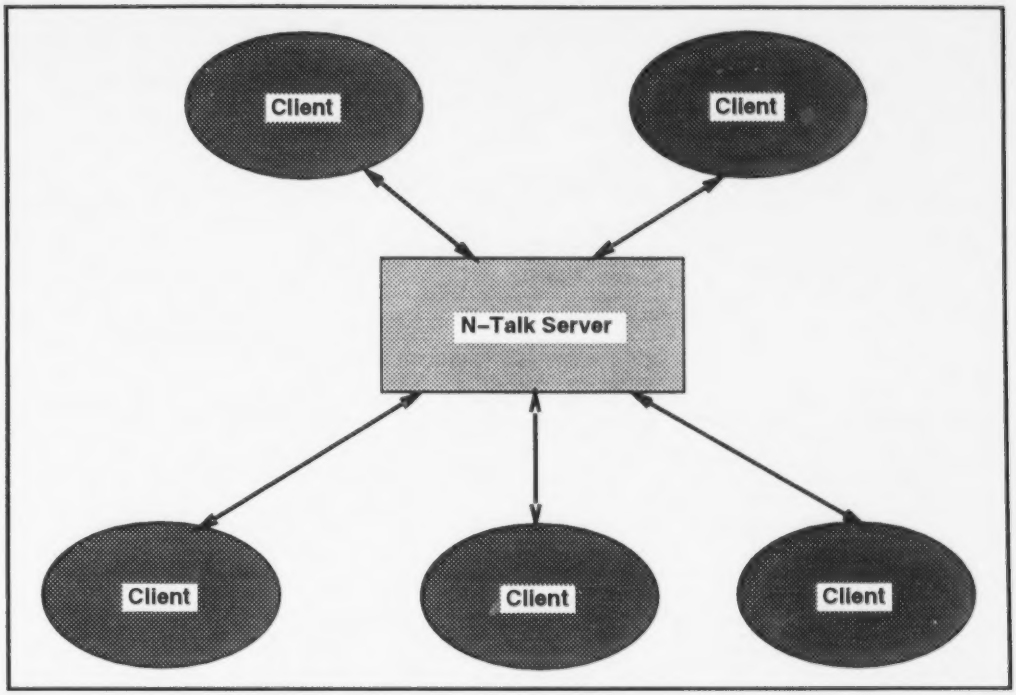

Figure 5.1: N-Talk Architecture

5.1.1 Features

N-Talk has the following features:

- create - request to create a session.
- join - request to join an ongoing session.
- invite - request to invite a user to an ongoing session.
- leave - request to leave a session that the user is a member of.
- terminate - request to terminate a talk session.
- ban - request to ban a user from joining a specified talk session.
- assign control - request to assign the leader's control right to a joined user for a specified session.

A collaborative talk session in N-Talk is created by sending a **Create Session** message to the N-Talk server. Figure 5.2 shows the **Create Ses** control flow. There are two types of talk sessions — *Formal* or *Informal* — which have several different characteristics. With an informal session, all participants have the same rights. A participant can add a user name to a ban-list to restrict someone from joining the session. Any participant in this session can remove the user's name from the ban-list to allow him/her to join the session. Participants can leave the informal session at any time. None of the participants can terminate the informal session except the last one. With a formal session, participation is by invitation only. The user who created the formal session becomes the leader of the session. Only the leader can add a user name to the ban-list to restrict someone from joining the session. The leader can grant a super right to other participants by adding them on a superuser list. The leader can release the join restriction by deleting the user's name from the ban list. The leader also can take back the super right from a superuser by deleting his/her name from the superuser list. Only the session leader can terminate the formal session. If other participants want to leave the session, they need to send a request to the leader. If the leader declines the request, they cannot leave the formal session. For fault-tolerance, we will randomly select one of the participants to take the leader role if the leader's machine crashes.

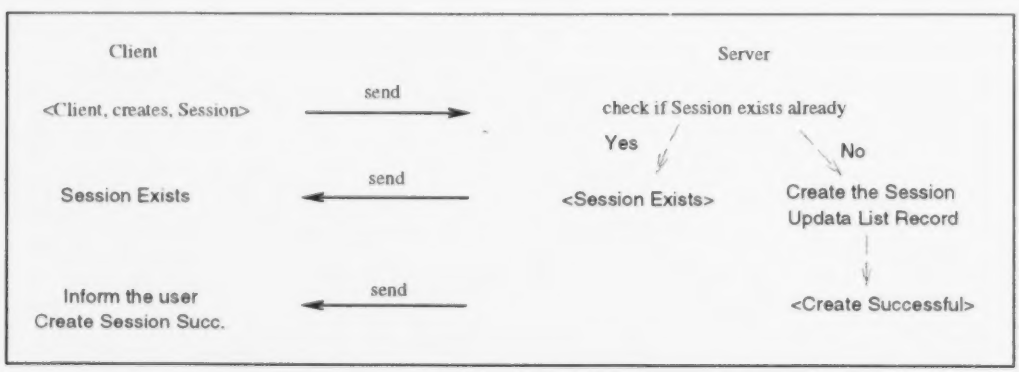

Figure 5.2: Create Session Control Flow

5.1.2 Interface

This section describes the user interface for N-Talk, which provides a number of functions:

- performs the N-Talk features through pushing a button.
- give a status report.
- give an information report.
- displays the talk contents.
- displays the participants' faces.

The N-Talk management window consists of three subwindows.

status subwindow: displays the action status information.

control subwindow: allows a user to perform the N-Talk functions.

information subwindow: performs and displays the sessions and users' information.

Figure 5.3 shows the N-Talk management window.

Status Subwindow

The upper left part of Figure 5.3 is the status subwindow. The status subwindow displays the status information. A user can know whether an action is successfully performed from this subwindow.

Control Subwindow

The control subwindow allows a user to control N-Talk. Through this subwindow, the user can perform all the functions provided by N-Talk. The left lower part of Figure 5.3 shows the control subwindow. Users can perform an action by selecting one of the buttons in this subwindow.

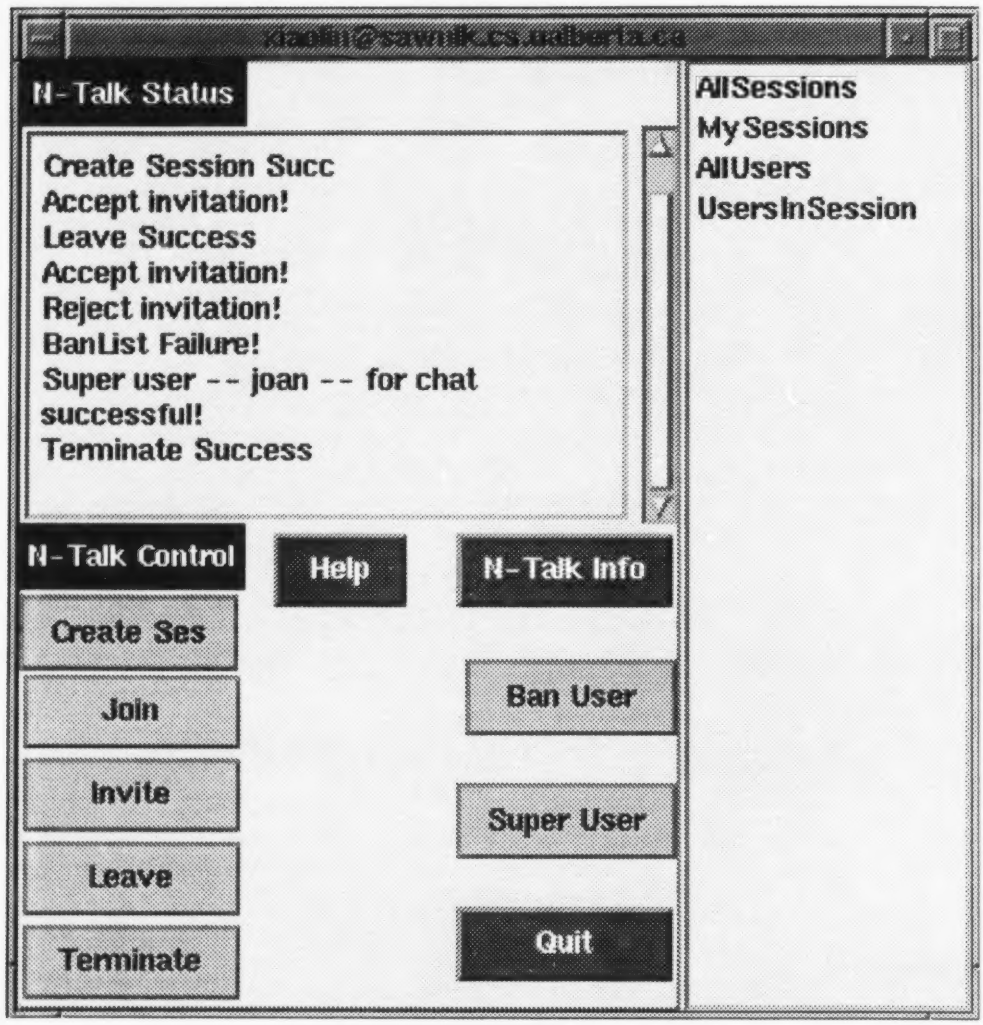

Figure 5.3: N-Talk Control Window

Create Ses — Once a user chooses *Create Ses* to create a new session, a pull-down menu will appear. The user can choose either *formal* or *informal* mode for the new session. Another window (Figure 5.4) is then displayed, prompting the user to enter a session name for the new session.

Join — Joins an existing informal session. If there are any sessions a user can join, a selection window (Figure 5.5) is displayed, and the user can select a session from this window. Otherwise, an information window will pop up, which tells the user that no session can be joined.

Invite — Invites other users to join an existing session. A selection window (Figure 5.5) will pop up after the user clicks the **Invite** button. An invitation window (shown in Figure 5.6) will be displayed in the invitee's machine. If there is no session to which the user can invite other users, an information window will pop up with this message.

Leave — Leaves an ongoing session. If the user joined several sessions, a selection window (Figure 5.5) will list all these session names and the user can choose the one which he/she wants to leave. If the session is a formal session, the leader will decide whether the user can leave. A request window (Figure 5.7) will pop up on the leader's machine.

Terminate — If a user chooses this button, a selection window will pop up and list all the sessions which the user can terminate. If there is no session which the user can terminate, an information window is displayed to tell the user this.

Ban User — Adds/Deletes a user name to/from the ban-list for a specified ses-

sion to restrict/allow the user to join the session. The session can be chosen from a selection window which is displayed after choosing Add/Delete action. The user name is typed in the prompt window (shown in Figure 5.8).

Super User — Adds/Deletes a user name to/from the superuser-list for a specified session to grant/take back the user's super right. The session can be chosen from a selection window which is displayed after the Add/Delete action is chosen. The user's name is typed in the prompt window (shown in Figure 5.8).

Quit — Quits the N-Talk service. A user should first leave all the sessions which he/she joined by using **Leave** or **Terminate**—then the user can quit the N-Talk service. For fault-tolerance, a question window (Figure 5.9) will pop up to confirm whether the user really wants to quit N-Talk. Otherwise, an error window (shown in Figure 5.10) is displayed to tell the user that he/she cannot quit N-Talk.

Help — Displays help on the functions of N-Talk.

N-Talk Info — Displays information about N-Talk.

Information Subwindow

The right part of Figure 5.3 is the information subwindow. This window integrates the command and display functions and provides a set of command lists that allow users to perform the following functions:

AllSessions — When a user selects this function, all existing sessions will be displayed in the information subwindow (see Figure 5.11). The user can return to the top level menu by selecting the *BACK* item.

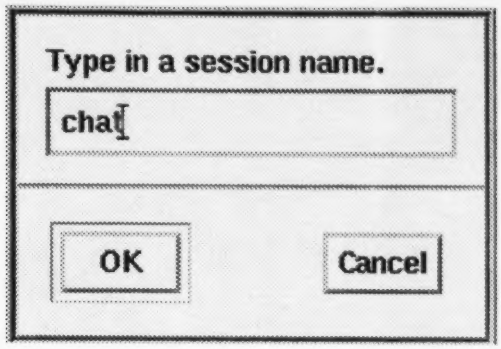

Figure 5.4: Session Name Prompt Window

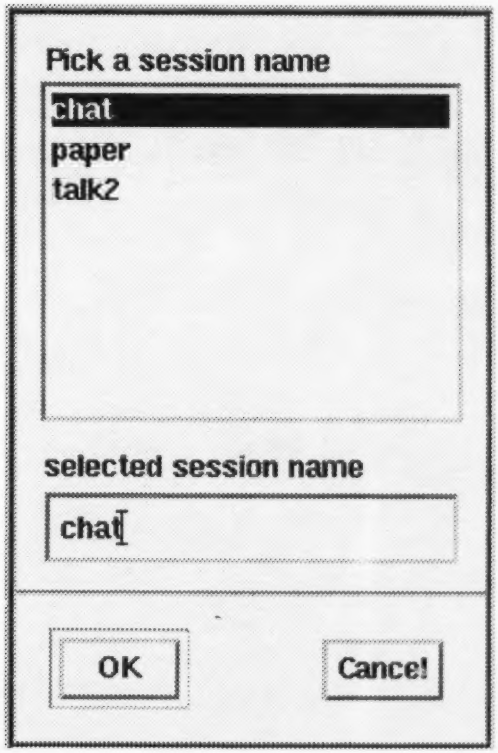

Figure 5.5: Session Selection Window

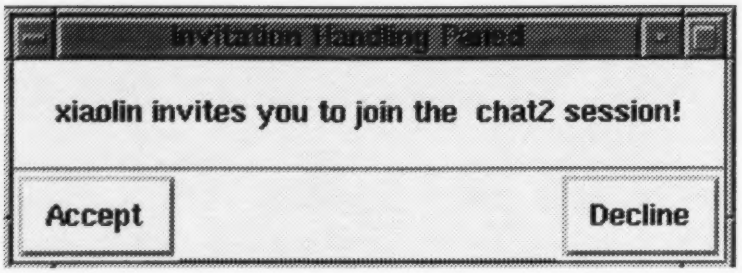

Figure 5.6: Invitation Window

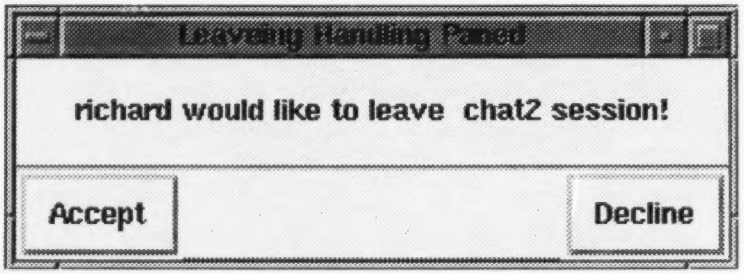

Figure 5.7: Leave Request Window

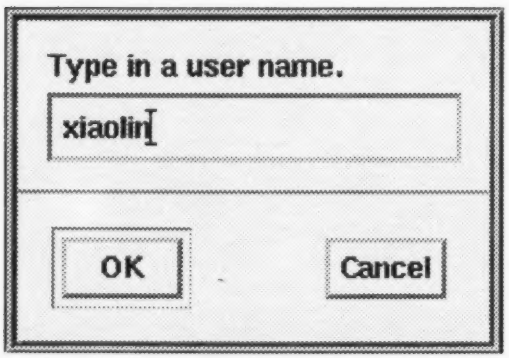

Figure 5.8: Username Prompt Window

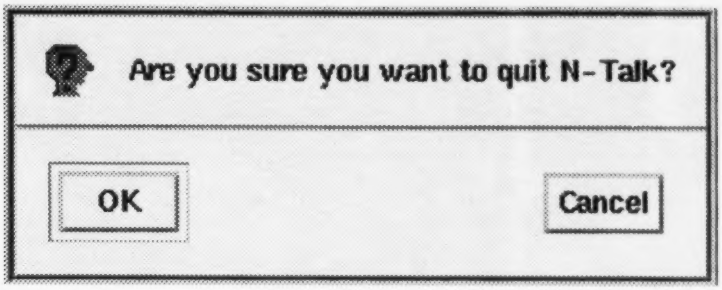

Figure 5.9: Quit Question Window

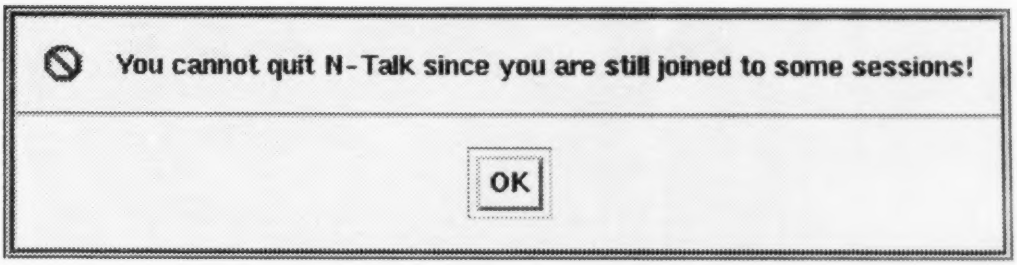

Figure 5.10: Quit Error Information Window

MySessions Displays the sessions which the user joined. The display method is the same as *AllSessions* function.

AllUsers Displays all the users who registered in N-Talk services. If a user wants to see the participants' pictures, he/she can select **AllFaces** to display all the participants' faces (Figure 5.12). One user's face can be displayed by selecting the user's name (shown in Figure 5.13).

UsersInSession Lists all users in a session. This function can be used to check who joined a specified session. As with *AllUsers*, participants' pictures can be displayed by selecting **AllFaces** or a user's name.

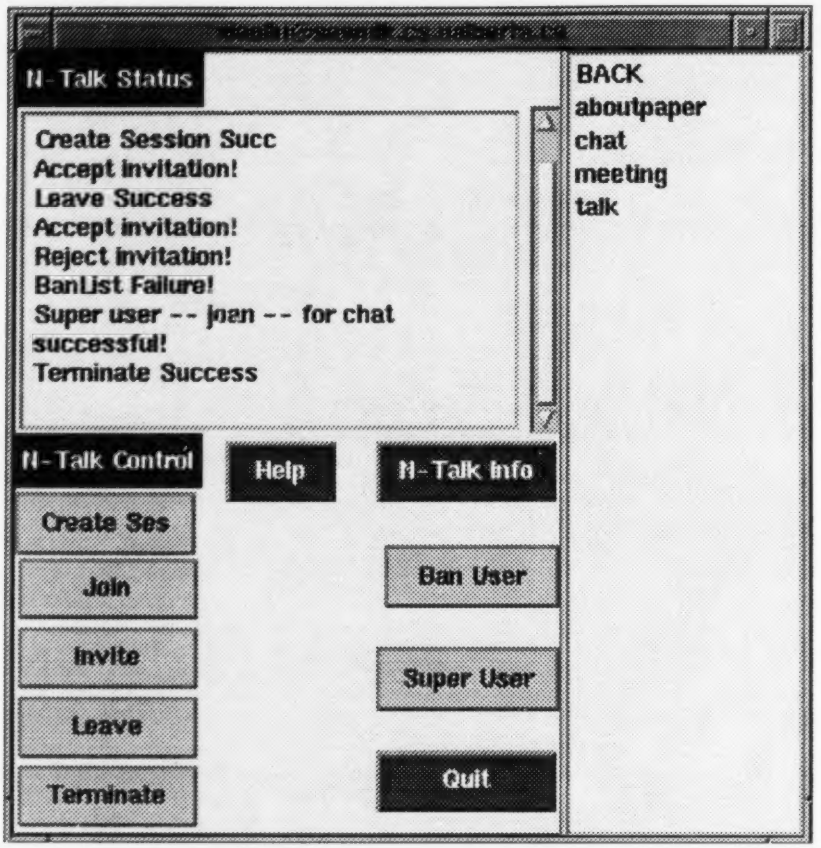

Figure 5.11: The Control Window Shows All Sessions

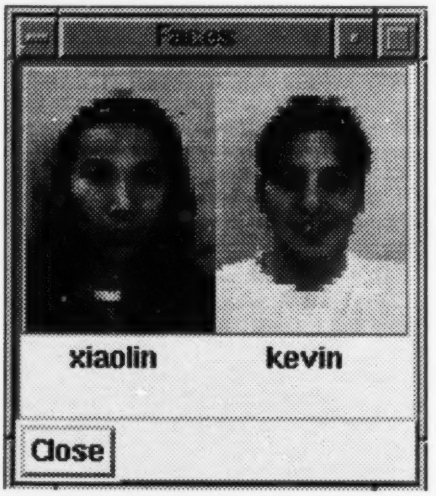

Figure 5.12: AllUser Faces

Figure 5.13: One User Face

Talk Text Windows

For each participant in a session, there is a window for him/her to display the text which he/she typed in. Figure 5.14 shows the text windows. A user can save the text by selecting the **Save** under **File** menu.

5.2 Implementation Details

The N-Talk service is implemented by using the **Client/Server** structure. It is a semi-replicated architecture, which is shown on Figure 4.3. The talk service was implemented on UNIX-based workstations. The client and server connect by stream socket.

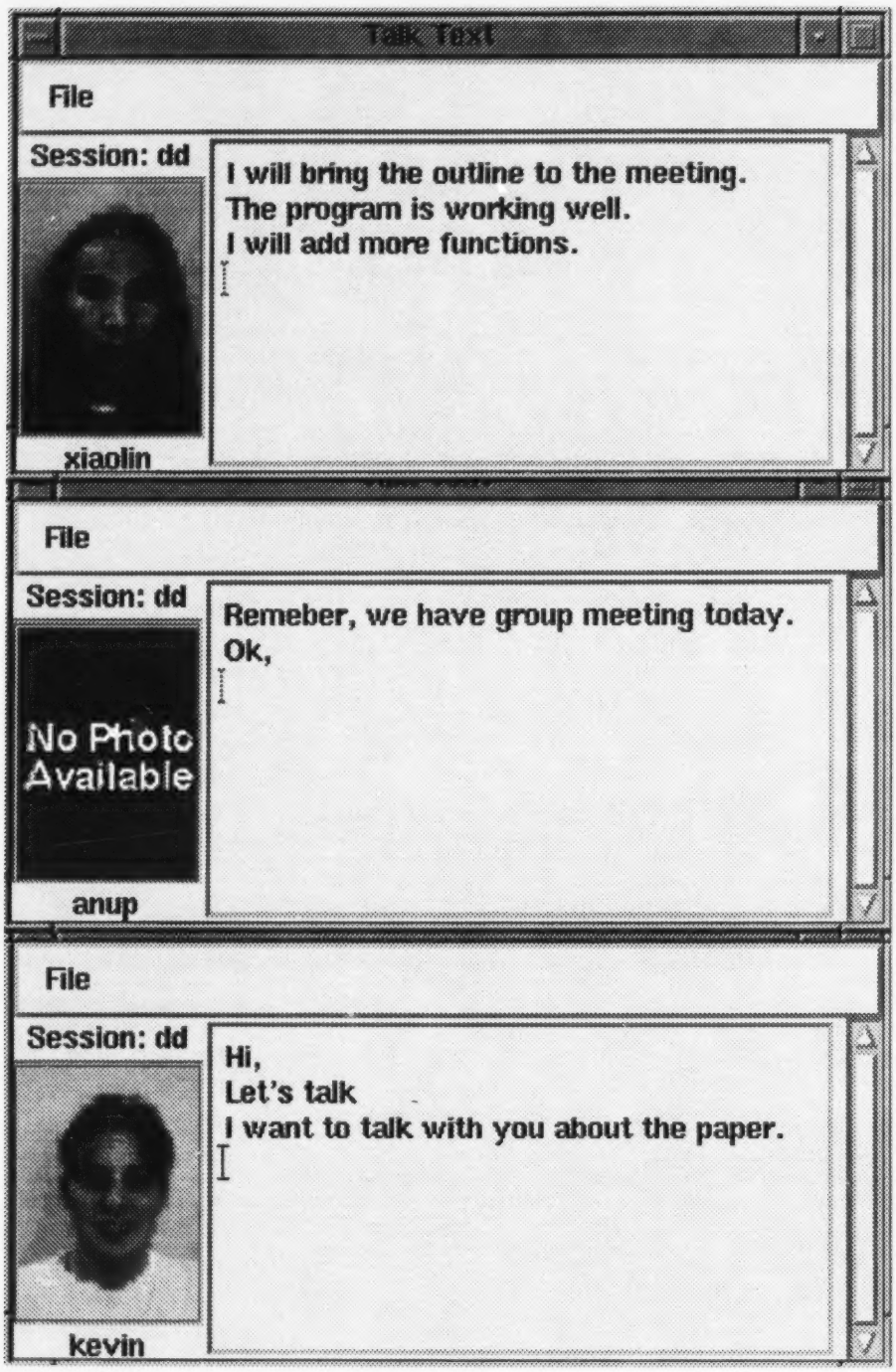

Figure 5.14: Talk Text Windows

5.2.1 Registration

Registration plays a very important part in collaboration system design. It provides important information to a registration user, such as who is available to invite and what meeting they can join. Roseman and Greenberg summarize several features which registration should provide to collaboration users[34].

- *Support registration of both people and their artifacts.* This collects the participants, equipment, and session artifacts. The information about participants helps people find who is available and what meeting they can join. The "equipment" is the collaboration application people use, such as drawing or talking. The "artifacts" are usually the documents.

- *Support the distribution registration information.* This information includes such things as who is around and their current activities.

- *Support a spectrum of group involvement in registration decisions.* The different sessions require registration of different group members according to different situations. For a formal session, only certain people can join, while everyone can join an informal session.

- *Support dynamic registration, so that people can be added or removed at any time.* In real life, people may join a meeting at quite different times. Some participants may want to leave the meeting before it is finished. Registration schemes should be flexible enough to allow people to join or leave sessions at any time.

In N-Talk service the registration information is maintained in the server site. Each client also maintains some information—such as its ongoing sessions. Almost all the points which we listed about registration are implemented in the N-Talk service. The registration in the N-Talk service presents the user with four types of lists:

- A list of all ongoing sessions.

- A list of all ongoing sessions which the calling participant has joined.
- A list of all users registered in the N-Talk service.
- A list of participating users for each ongoing session.

5.2.2 Motif User Interface

The user interface is implemented by using **Motif** for Sun workstations. The reason for using Motif to implement the interface is that it is one of the major interface standards in the UNIX world. The interface is responsible for all user interactions, and contains all callback functions which handle all the window events. A callback can be invoked by a user who performed an action that triggers its call.

5.2.3 N-Talk Client

The communication system on the client side is responsible for sending commands and receiving the results from the server. The client and server connection is implemented by using the UNIX *socket* facility. A client sends or receives a message by using the message transfer primitives. The message transfer primitives are listed in Section 5.2.5.

On the client side, there are several data structures which record the important information for handling N-Talk sessions and users' talk text windows. For example, The Talking_Member structure records a member's name and his/her talk window information. With this information, we can display a user's talk context in an appropriate window according to the user's name. The relevant data structures are:

- **Session:** is a link structure which records the session information. This includes the session name, session mode (informal or formal), session type (such as talk, image), leader name, a pointer to the members of the session, and pointers to other sessions.

```
typedef struct Session {        /* Struct for session */
```

```
    int mode;
    int type;
    char name[SESSION_NAME_LEN];
    char leader[USER_NAME_LEN];
    talking_mbr_type *mbr_head_ptr;

    struct Session *s_link;
    struct Session *s_rlink;
};
```

- **Talking_Member:** records the member information. It includes username and win_no. It is a two direction link structure. It includes two pointers to the left and right neighbor. The *win_no* item records the window ID since this is needed for window update.

```
typedef struct Talking_Member {
                    /* Structure for a member of a session */

    char username[USER_NAME_LEN];

    int  win_no;    /* Window information */

    /* links */
    struct Talking_Member *m_link;
    struct Talking_Member *m_rlink;
  } talking_mbr_type;
```

- **Face Struct:** is a structure for face display. It includes image size, a pointer to the face image, and the person's name.

```
typedef struct
    {
      int size;
      char *image;
      char name[USER_NAME_LEN];
    } face_struct;
```

5.2.4 N-Talk Server

The server is implemented by using C programming. We use the UNIX *socket* facility to implement the connection between clients and server. The server also uses the message transfer primitives (see Section 5.2.5) to send/receive messages to/from clients.

It consists of the following main data structures.

- **Register:** maintains the register information. When a new user registers in N-Talk, related information will be recorded and the new user added to the register link.

```
typedef struct reg_name {      /* struct for the registered name */
   int cont_sock;                      /* control socket number */
   int data_sock;                      /* data socket number */
   char realname[USER_NAME_LEN];       /* user login name */
   char clientname[USER_NAME_LEN];     /* user nickname */
   char hostname[HOST_NAME_LEN];       /* machine name */
   struct ses_link *my_ses_head_ptr; /* user's sessions */
   struct reg_name *u_link;            /* left link */
   struct reg_name *u_rlink;           /* right link */
 } reg_name_type;
```

- **Banlist:** is a structure to record the banned users. If a username appears in the banlist for a specified session, that user cannot join the session.

```
typedef struct banlist {        /* struct for ban list */
   char name[USER_NAME_LEN];
   struct banlist *b_link;
   struct banlist *b_rlink;
} banlist_type;
```

- **Superlist:** is a structure responsible for recording the super users.

```
typedef struct superlist {      /* struct for super right list */
   reg_name_type *u_ptr;
   struct superlist *su_link;
   struct superlist *su_rlink;
} superlist_type;
```

- **Session:** handles the session control. It is one of the main data structures. Most of the actions need to access the data structure.

```
typedef struct Session {        /* Struct for session */
   char name[SESSION_NAME_LEN];     /* session name */
   int type;                        /* session type */
   int mode;                        /* session mode */
   talk_mbr_type *mbr_head_ptr;     /* session member pointer */
   char leader[USER_NAME_LEN];      /* session leader name */
   superlist_type *super_head_ptr;  /* superlist pointer */
   banlist_type *ban_head_ptr;      /* banlist pointer */
   struct Session *s_link;          /* left link */
   struct Session *s_rlink;         /* right link */
} talk_ses_type ;
```

The relationship between these data structures is shown in Figure 5.15.

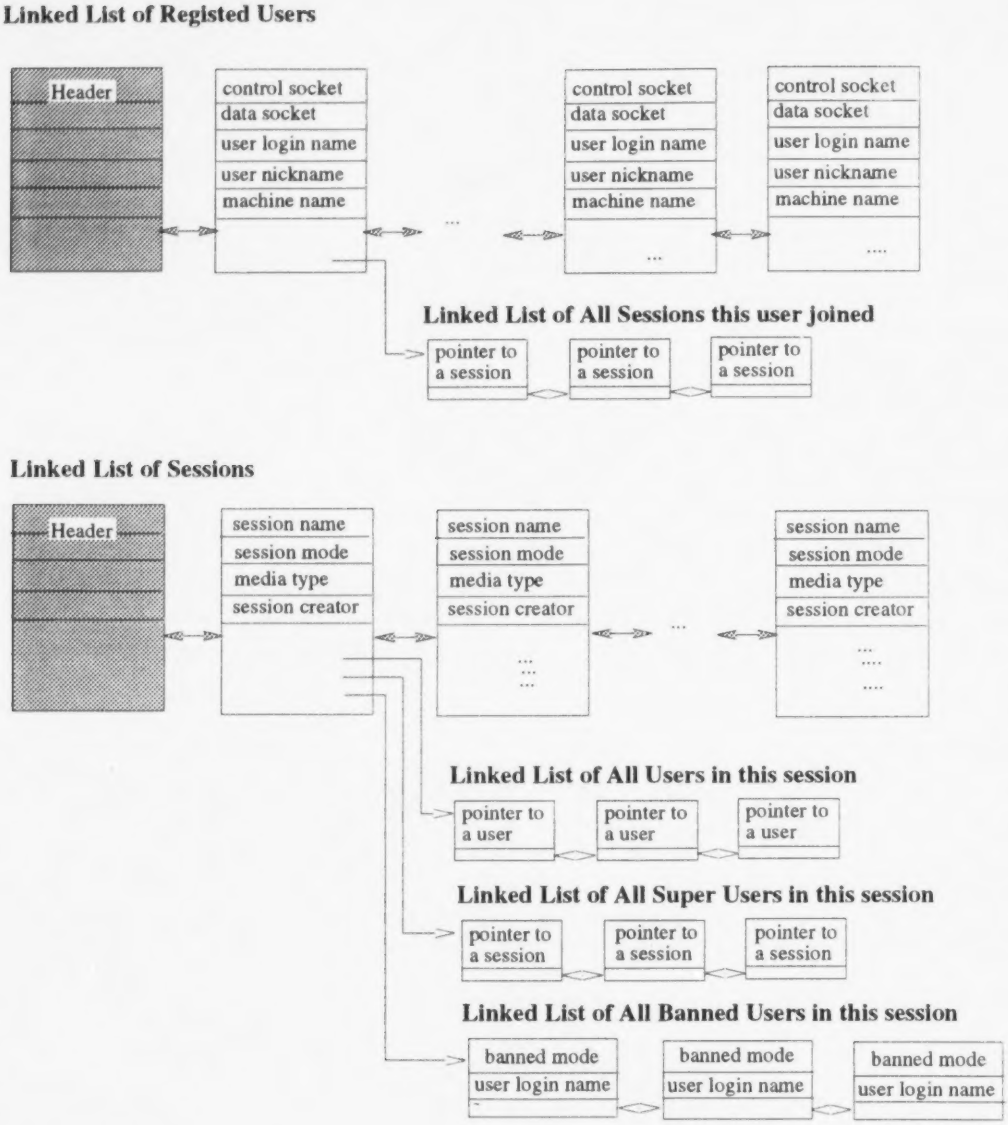

Figure 5.15: Data Structures

5.2.5 Message Transfer Primitives

A set of package and unpackage primitives for message transfer between client and server is implemented. The reason for implementing these primitives is that there are several different kinds of messages. We wanted to find a flexible way to send and receive a message, and the primitives provide a method to deal with this. The sender can package a message by choosing some of the primitives and putting them in any order, depending on the needs of the sender. The receiver must unpackage the received message in the same order in which it is packaged. Table 5.1 lists these message transfer primitives.

5.2.6 Concurrency Control

Figure 5.14 shows the talk text window. Each participant has a separate window to show his/her text in each site. There is no need to have a concurrency control scheme in the talk communication service.

5.3 Generalizing Design

The above implementation experience, although demonstrated with the N-Talk service, is extendable and applicable to other distributed real-time collaboration application systems/tools. Actually, the message transfer primitives are used in other tools. The servers were implemented in almost the same way for N-Talk, Co-Draw and Co-Image. The registration scheme can be used for the drawing application, the image display, and manipulation application, etc. The N-Talk service provides very good session management. It is easy to get an overview of an existing session, to join an ongoing session, or to create a new session. The session management can also be used for other applications.

initsend(char *buffer)	Initialize the send buffer
initrcv(char *buffer)	Initialize the receive buffer
longIntpk(long number)	package a long integer number
longIntunpk(long *number)	unpackage a long integer number
intpk(int number)	package an integer
intunpk(int *number)	unpackage an integer
virtual_intpk(int number)	virtual package an integer
real_intpk(int *number)	real package the integer which is packaged by virtual_intpk
namepk(char *name, int size)	package a char string
nameunpk(char *name)	unpackage a char string
textpk(char *text, long size)	package a text string by using byte copy
textunpk(char *text, int size)	unpackage a text string
imagepk(char *image, int size)	package an image
imageunpk(char *image, int *size)	unpackage an image
endsend(char *buffer)	end package the sending message
endrcv(char *buffer)	end unpackage the received message
get_message_size(char *buffer)	get the size of a message

Table 5.1: Communication Primitives for N-Talk

Chapter 6

Integration of Image Processing and Text/Picture Editing

Great interest has developed in recent years in building tools which allow people to collaborate on work without the need for physical proximity. In this chapter, several collaborative tools which are integrated into our system will be described.

6.1 Co-Draw: A Collaborative Drawing Tool

6.1.1 Overview

Tele-presence is a way to give distributed participants a feeling that they are in the same location. Real-time meeting support applications belong in this field. The goal of these systems is to transmit both the explicit and subtle dynamics that occur between participants. These include body language, hand gestures, eye contact, knowing who is speaking and who is listening, voice cues, focusing attention, and so on. Tele-presence facilitates effective management of remote meetings by using the natural and practised techniques utilized in face to face meetings. Tele-data is a method to allow distributed participants at a session to present or access physical materials that would normally be inaccessible to the distributed group. These include

notes, documents, plans and drawings. The network computer has become a valuable medium for people to share on-line work with each other [20]

Co-Draw is a collaborative drawing tool in the field of tele-data. It supports a small group (around 10 people) with real-time access to a shared drawing space. The multi-users are equal.

Normally, a group process begins with a set of initial design meetings. The group members express, discuss, and develop ideas in the meetings. Participants typically use some large communal work surface to facilitate their interactions. The work surface is a group drawing space which includes whiteboards, blackboards, large sheets of paper, and a variety of pens or chalk. The Co-Draw tool provides a remote whiteboard work surface to a group of people.

The Co-Draw main features are:

- a what you see is what I see (WYSIWIS) display; user actions are immediately visible on all screens
- support of latecomers
- simultaneous interaction is fully supported
- a good mixture of graphics and textual work space

6.1.2 Implementation Details

Co-Draw is built upon the UNIX platform. It runs on Sun workstations connected by a network. The distributed architecture of Co-Draw is semi-replicated architecture, with the participant process running as a single process on every workstation. There is no special concurrency control policy used in the Co-Draw tool. In spite of its simplicity, however, Co-Draw is effective. The response time for Co-Draw is small, since only the input is sent between the participants. Participant processes communicate via UNIX stream sockets using several events which are listed in Table 6.1.

Event	Information passed
register a new user	host name, user login name or nickname
leave	user login name or nickname
drawing an item	item shape, first item coordinates, the width and height of an item or the second item coordinates, text string
image transfer	binary date of the work surface image
undo an action	
clearing screen	

Table 6.1: Communication Protocol between Processes

Co-Draw provides several types of items. There are points, lines, rectangles, ovals, filled rectangles, filled ovals, and text. The information which an item passes depends on the type of item, as shown in the following table.

Shape	Information passed
point	item shape, coordinates of the point
line	item shape, start and end coordinates
rectangle	item shape, start coordinate, width and height
oval	item shape, center coordinate, width and height
text	item shape, start coordinate, text string

Table 6.2: Detailed List of Items

A participant can join an ongoing session at any time. The participant connects with the drawing server and sends his/her login name, as well as host name, to the server. The server sends an image transfer request to one of the existing participants to get the current state of the shared work surface, and then sends the image to the newcomer. This program can also support the single user case. This means a user can use the Co-Draw tool even if there are no other participants.

A participant can choose an appropriate item shape depending on his/her specific drawing need. The tool includes a good mixture of graphics and textual context on the shared work surface.

Figure 6.1 shows the shared work surface for Co-Draw. The two windows are exactly the same. It is a strict WYSIWIS environment.

6.2 Co-Image: An Image Display and Manipulation Tool

The work presented here centers on multiuser image display and interactive manipulation. This discussion includes how to implement a networked, multi-participant, wide-area, on-line image display and manipulation tool.

6.2.1 Functionality and Features

Images exist everywhere. Image display and manipulation tools are used in almost all disciplines. An image and manipulation tool not only displays images, but also performs some functions on the images. Most of these kinds of tools support just one user, however, the idea of shared applications in the computing environment has been around for a long time. Co-Image is an application tool by which single-user applications using the MIT X11R5 windowing protocol may be shared among many users. It allows users at multiple sites to collaborate by viewing and manipulating the output simultaneously. Many may watch one user manipulate the tool, and can

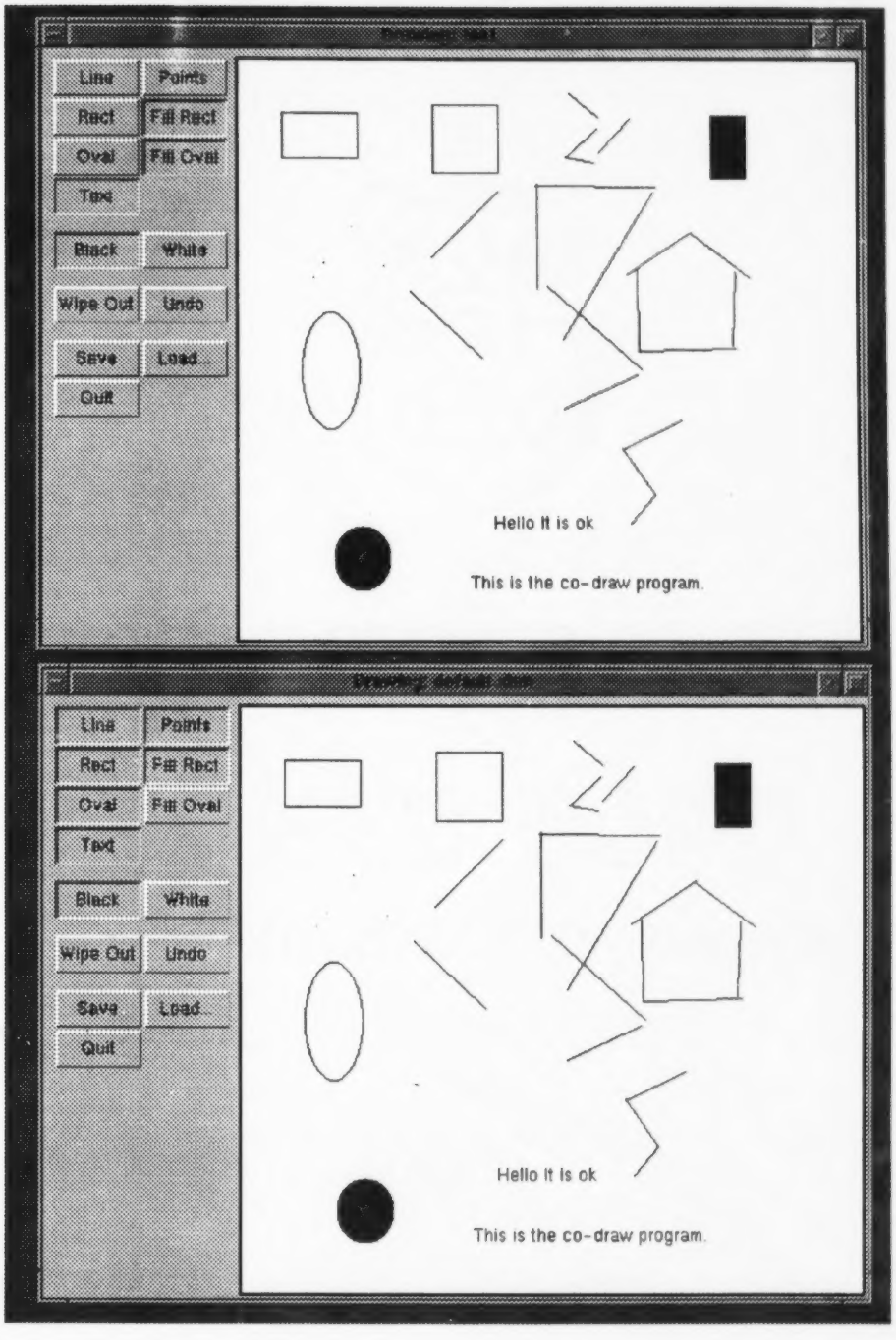

Figure 6.1: Co-Draw Windows

suggest or execute alternative methods or corrections, all from sites that may be virtually anywhere.

A collaborative image display and manipulation tool, Co-Image, can perform these functions on the image:

- copy a region of the image
- paste a region of the image
- cut the image
- crop the image
- resize the image
- trim the image edges
- gamma correct the image
- edge detection
- change image size
- rotate the image
- vary the color brightness
- add a border to the image
- oil paint an image

The Co-Image main features are:

- WYSIWIS display
- support of pessimistic locking concurrency control policy

6.2.2 Implementation Experiences

Distributed Architecture

As we mentioned before, a single copy of the application exists in a centralized architecture. The central server distributes output to all session sites. If a centralized architecture is used in the Co-Image display and manipulation tool, a high bandwidth will be required since the server needs to send the whole image to client sites. The response time will be long. If a copy of the application runs at every site in the session, only input needs to be sent. For example, one participant cuts a region of an image. Only the start point coordinates, the width and height of the region, and the operation command need to be sent to every site—instead of the updated image. The bandwidth will be low and the response time is reduced. Considering that many functions, such as paste, are two-step operations in the display and manipulation tool, the Co-Image requires concurrency control.

Co-Image uses a semi-replicated architecture. The semi-replicated architecture has three advantages:

- *Performance* First, it requires lower bandwidth because only input must be distributed among the sites. Second, it is less sensitive to variations in network latency; all participants in the session receive good interactive performance since they interact with local copies of the tool.

- *Concurrency control* The simplest way of implementing concurrency control is through a centralized architecture. The semi-replicated architecture has this advantage because of the centralized server.

- *Support customizable local view* Like replicated architecture, the semi-replicated architecture handles local management of the view which means that different views are easily supported. Each participant can tailor his/her display of an image without affecting the display of other participants.

Concurrency Control

Because some functions are two-step operations, concurrency control is necessary. There are several methods for handling concurrency control, such as the pessimistic approach, fully-optimistic locking, and semi-optimistic locking.

We use the *single active participant* mode to handle the concurrency control in the Co-Image tool. It is a pessimistic approach. Only one participant at a time can do some operations in the shared image. If a participant wants to do an operation on the shared image, he/she must get a lock first. The centralized server is responsible for concurrency control.

The original image is shown in Figure 6.2.

Figure 6.2: The Original Image

Figure 6.3 shows the updated images after one participant performed the cut and paste functions.

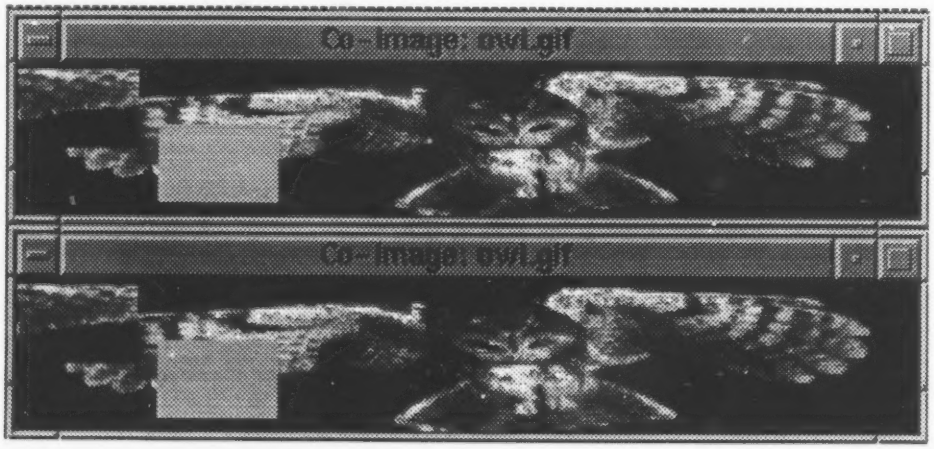

Figure 6.3: The Updated Images after cut and paste

What do computer displays do about gamma? Many computer displays ignore the effect of monitor gamma. The frame buffer value provided by the application software is converted linearly into voltages that drive the CRT in the display. The values in the frame buffer are not proportional to the resulting brightness. A frame buffer value of 1/2 the maximum will produce less than 1/2 the brightness. The Figure 6.4 is the corrected image of the original one at a gamma of 1.8.

Figure 6.4: The corrected image

Figure 6.5 shows the images after one of the participants used the gamma correction function. The gamma value is 2.8.

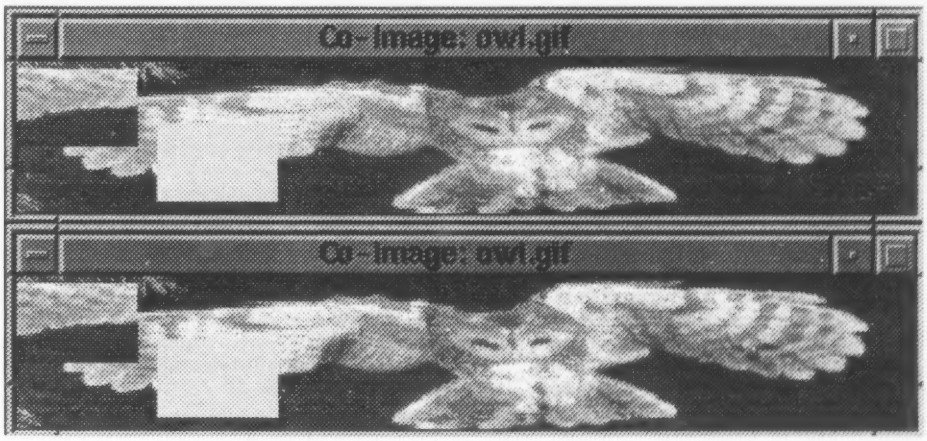

Figure 6.5: The updated images after gamma correcting the image

6.3 Co-Edit: A Collaborative Edit Tool

6.3.1 Overview

People working together to produce a single document is a common occurrence— both in the business and research worlds. In the sciences, the proportion of co-authored articles has been increasing steadily. Lubich used the following statistics to show the dramatic increase in internationally co-authored articles[5][28]. In some fields, over 65% of articles are jointly written.

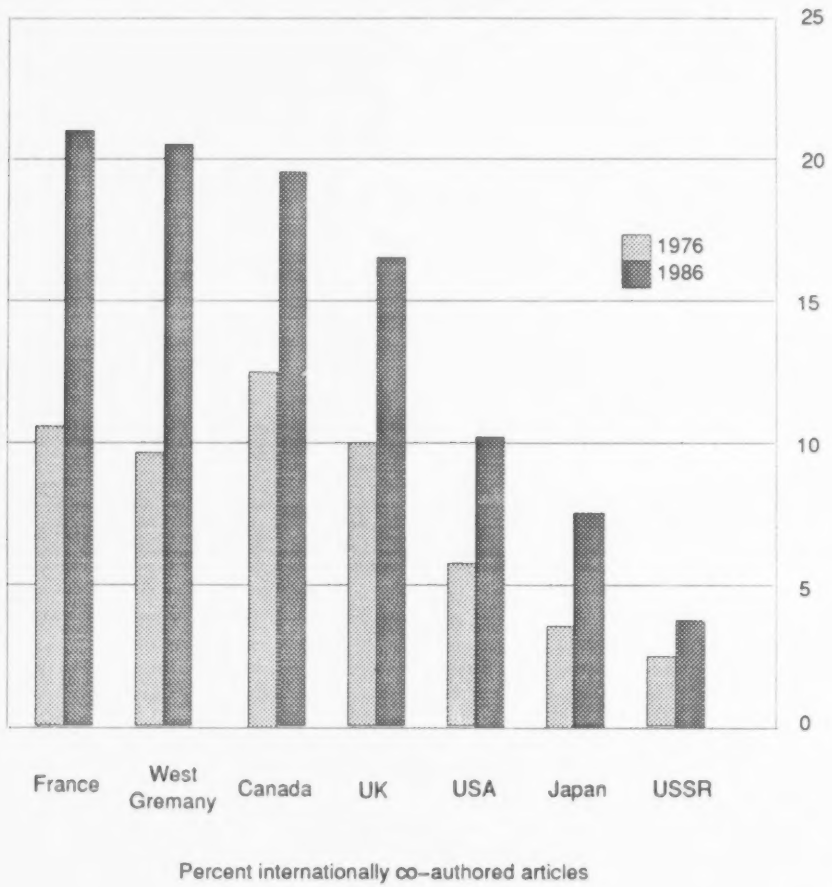

Figure 6.6: Increase in internationally co-authored articles

Co-authors need to share information and to coordinate activities. Unfortunately,

current word-processing tools, such as vi, emacs, textedit, and xedit, do not support these functions. Consider two authors working jointly on a paper. The two authors would like to work in parallel on the same paper from the workstations in their offices. In this case, the two authors have to work on different aspects of the paper. Sometimes, they might need to reorganize or rewrite a paragraph. As we know, many significant problems in text (e.g., voice, persuasiveness, organization), though easy for an experienced writer to detect, cannot be easily described. For such problems, rewriting is often a more efficient strategy than trying to diagnose the problem, and writers often choose this strategy when revising others' texts. It is difficult to do this without a co-edit system. However, a collaborative editing system allows multiple users to view and edit a shared document simultaneously from geographically different sites.

6.3.2 Requirements for Collaborative Writing

The requirements of a collaborative editing system are quite simple:

- *Hiding of communication protocols:* adapting an editor to a group editing system should not require knowledge of distributed systems issues.
- *Multi-user collaboration:* the system allows users to collaboratively edit a shared document from different places.
- *Support single user:* adding collaboration support to a writing project should not affect an author's ability to write an individual document.

6.3.3 Distemacs Co-Edit Tool

There are several existing co-authoring systems, and so there is no reason to implement another co-authoring system rather than using an existing system. After comparing some co-authoring systems, we integrated the **Distemacs** Co-Edit tool [27][31] to our system. **Quilt**[16] and **PREP** [29] are asynchronous collaborative authoring

systems. They use roles, explicit annotation, and structured or directed messaging to provide the means for generating awareness and coordination information. However, since they are asynchronous systems, they do not satisfy our requirements for collaborative writing. **Grove** [14], **ShrEdit** [13] and **MACE** [30] are synchronous collaborative writing systems which support multi-user work on a shared document. However, these applications do not provide effective and flexible undo facility. **Distemacs** Co-Edit tool provides a synchronous collaborative writing function. It is designed for using in both face-to-face and remote collaborative situations. This tool was developed by Professor Atul Prakash and his group members at the University of Michigan. **Distemacs** was implemented by using **DistEdit** toolkit. Distemacs has the following features [26]:

- allows several users to edit the same file simultaneously in a single session.
- provides support for locking of regions.
- allows users to undo the globally last as well as their own last actions.
- provides a window to monitor and control the group session.

In section 6.3.1, we gave a two authors case. In this case, the two authors can edit the same paper at different times by using **Distemacs**. Suppose one of authors invokes his/her editor on the shared file, and then sometime later the second author wants to edit the same file. He/she just uses the same procedure as with a single-user editor to invoke the **Distemacs** editor. A joint editing session is established since the two authors edit the same file. **Distemacs** ensures that they will have consistent views of the file being edited at all times. Each user may edit or view a different portion of the document and could have a different window size, as illustrated by the following two figures. Figure 6.7 shows the sample screen in such a situation. The DE-session window is shown in Figure 6.8.

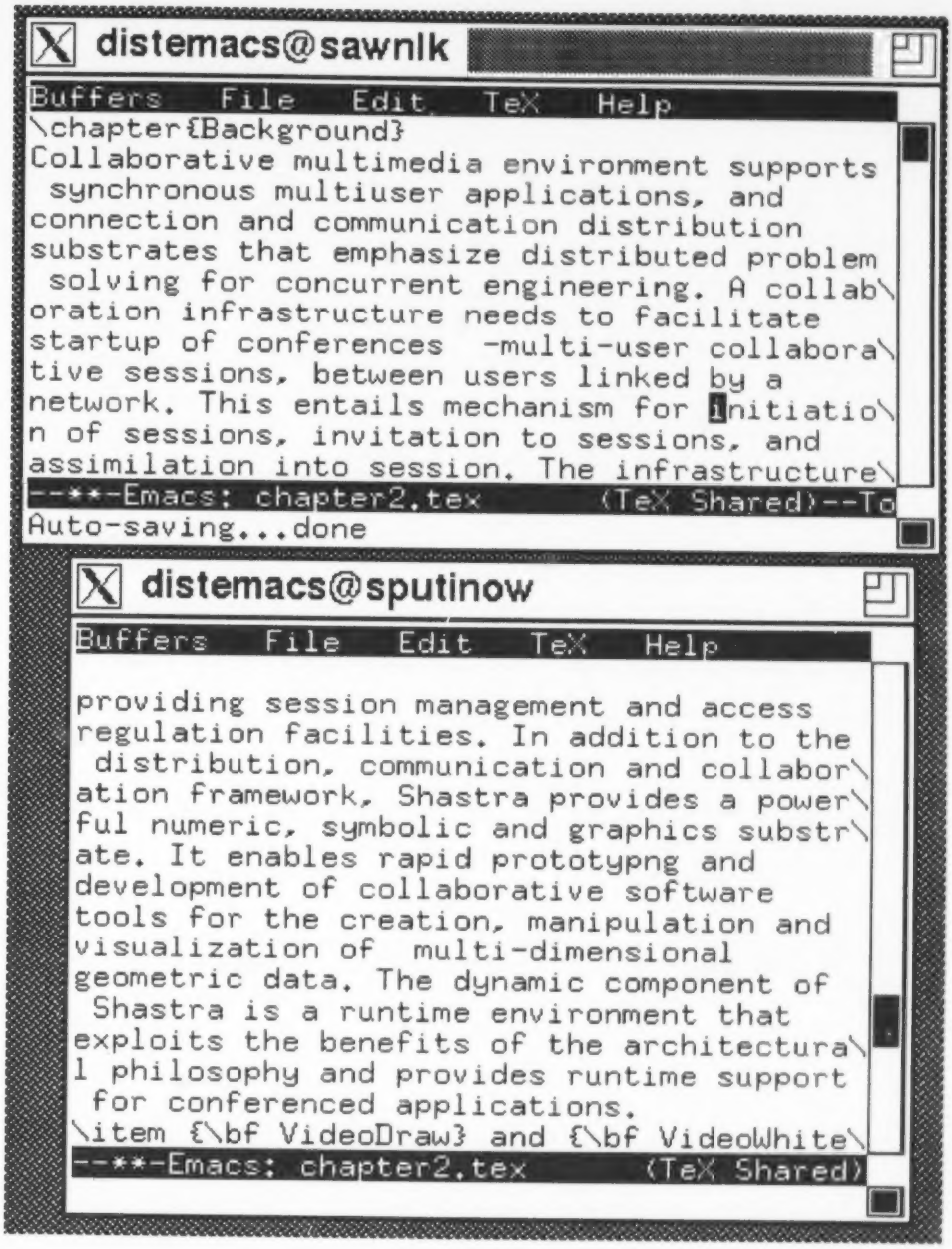
distemacs@sawnlk
Buffers File Edit TeX Help
\chapter{Background}
Collaborative multimedia environment supports
synchronous multiuser applications, and
connection and communication distribution
substrates that emphasize distributed problem
solving for concurrent engineering. A collab\
oration infrastructure needs to facilitate
startup of conferences -multi-user collabora\
tive sessions, between users linked by a
network. This entails mechanism for initiatio\
n of sessions, invitation to sessions, and
assimilation into session. The infrastructure\
--**-Emacs: chapter2.tex (TeX Shared)--To
Auto-saving...done
distemacs@sputinow
Buffers File Edit TeX Help
providing session management and access
regulation facilities. In addition to the
distribution, communication and collabor\
ation framework, Shastra provides a power\
ful numeric, symbolic and graphics substr\
ate. It enables rapid prototypng and
development of collaborative software
tools for the creation, manipulation and
visualization of multi-dimensional
geometric data. The dynamic component of
Shastra is a runtime environment that
exploits the benefits of the architectura\
l philosophy and provides runtime support
for conferenced applications.
\item {\bf VideoDraw} and {\bf VideoWhite\
--**-Emacs: chapter2.tex (TeX Shared)

Figure 6.7: Two People Using Distemacs Editor simultaneously

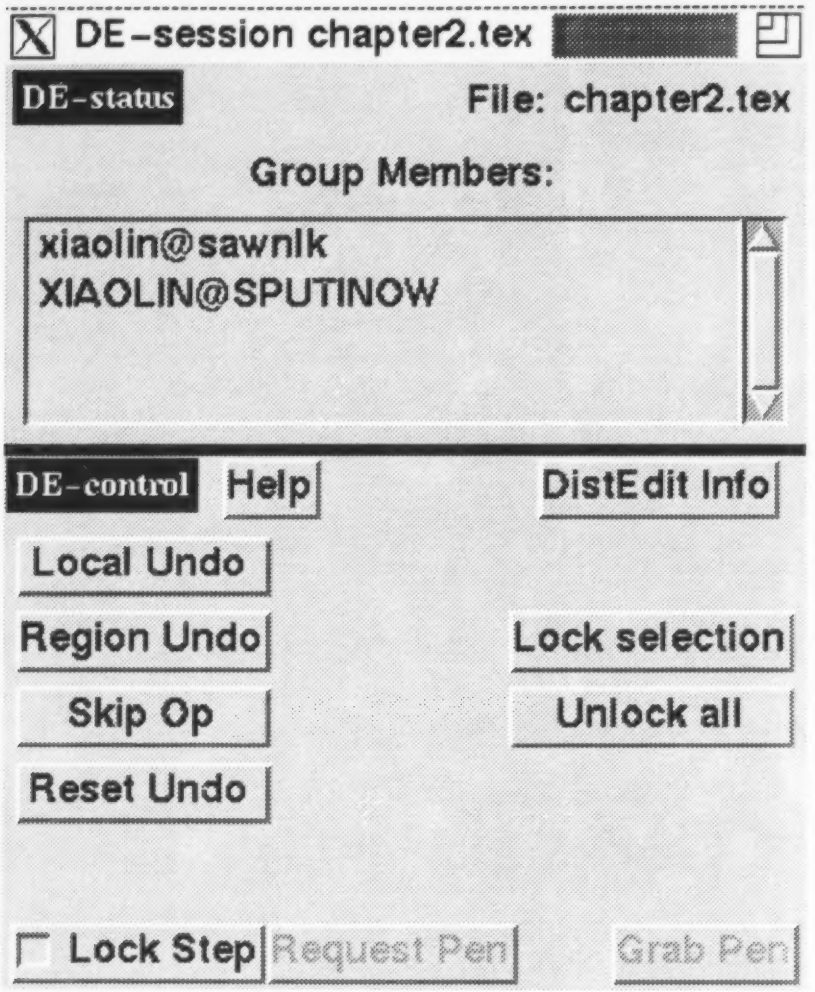

Figure 6.8: Distemacs Control Window

The group editor should support concurrency control to prevent two or more users from simultaneously trying to edit at exactly the same place in the shared document. *Automatic locks* are designed to handle this. When a user does any editing, a temporary lock is acquired automatically on the part of the document to be modified. In Distemacs, the editor will automatically lock the current line which the user is editing. It is a distributed version of the GNU Emacs.

DistEdit is a toolkit that can be used to build new group editors and adapt existing single-user editors to the task of group editing [27]. **DistEdit** provides a set

of primitives that can be used to add collaboration support to existing text editors with minimal changes to their code. **DistEdit** not only supports *automatic lock* function but also provides *explicit locks*. If one user lock a region, the other users cannot alter that region until the lock is released.

Knister and Prakash give the following two figures, Figure 6.9 and Figure 6.10, in their papers [26][27]. Figure 6.9 shows the structure of a typical single-user text editor. The structure of **DistEdit**-based editors is shown in Figure 6.10.

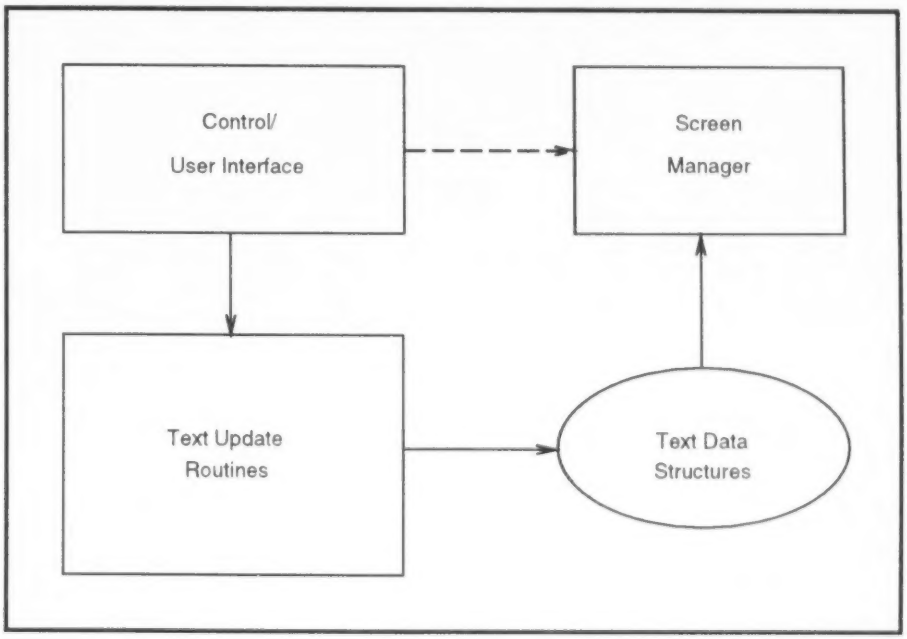

Figure 6.9: Typical structure of a single-user editor

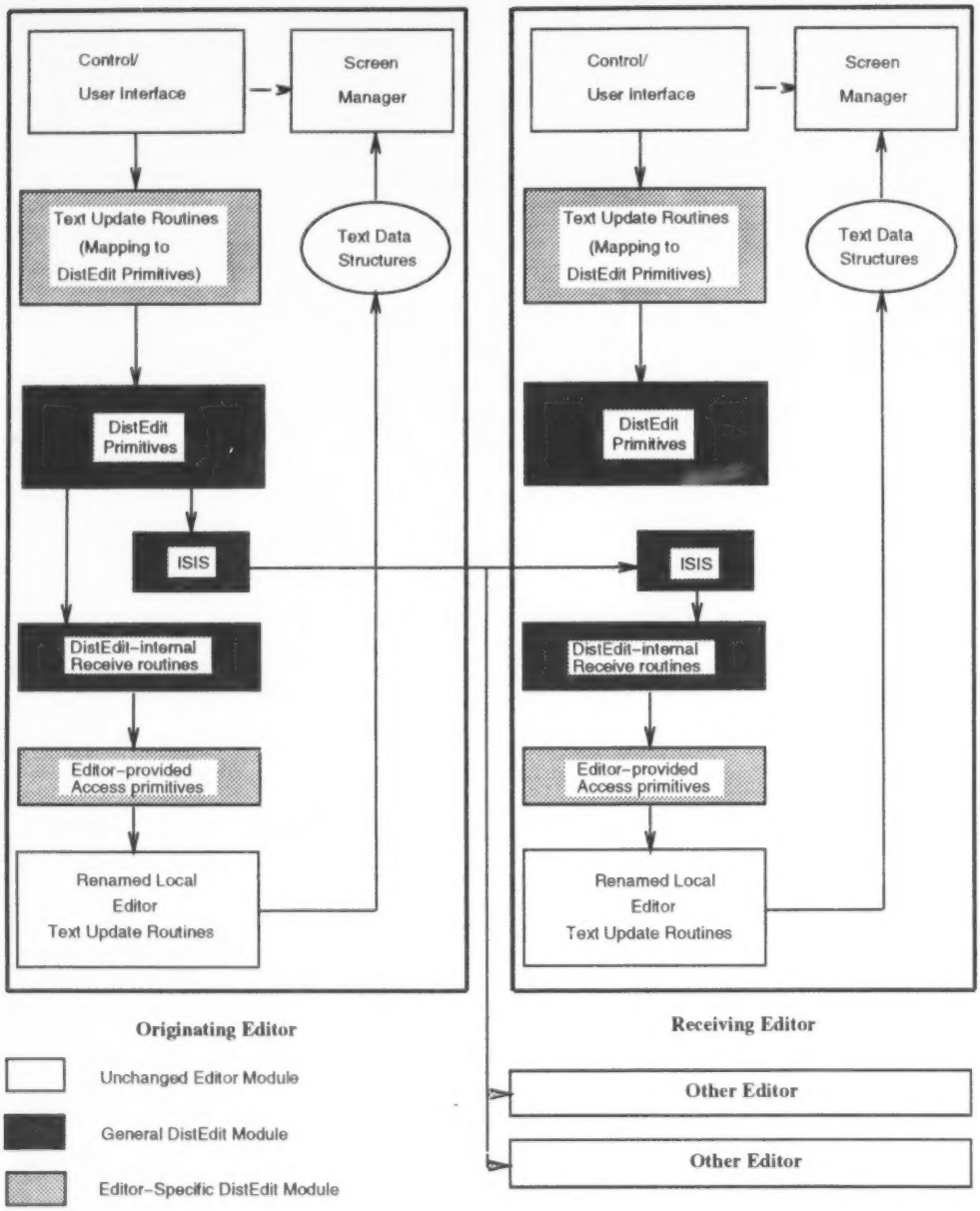

Figure 6.10: Structure of an Editor built using the DistEdit toolkit

ISIS [7] is a toolkit for programming distributed applications. It provides elegant broadcast facilities. **DistEdit** uses **ISIS** as its communications package.

We integrated the **Distemacs** into our collaborative system. Our other tools provide several methods to joint authors to let them communicate. They can discuss their joint work by using N-Talk, and they can share-draw some illustrations for their paper.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis describes a collaboration system, built on UNIX workstations, to allow a group of people distributed geographically to collaborate on a project. The various issues which are involved in collaboration systems are discussed, and other work in this research area is reviewed. The system which we have built provides talking, drawing, image processing, and co-authoring capabilities. This is a real-time collaboration application—however, the design principles can be extended and applied to other multimedia collaboration applications such as audio/video conferencing systems, collaborative CAD systems, and tele-learning applications.

The main contribution of this work is the development of a collaborative system which supports the multi-user: talking, drawing, image processing, and co-authoring.

The features of this work are:

- Effective session management. It is easy to get an overview of an existing session, to join an ongoing session, or to create a new session.
- Support of different level interaction coupling, such as strict WYSIWIS and loose WYSIWIS displays between participants.
- Support of single user. This means the collaboration application system works

in the one participant case.

- A friendly interface that relieves users from a lot of typing. In many functions users just need to make a selection. This also reduces the mistakes which are the result of typing errors.

Motif was used for implementing the user interface. **Motif** provides an effective tool for building the graphical user interface.

7.2 Future Work

The work presented here should be seen as an initial step in investigating the development of real-time collaborative application systems. Thus, the real-time collaboration system can be enhanced in a variety of ways.

- Currently, Co-Draw and Co-Image support only one session. This means that all users participate in the same session. All participants draw on only one shared space, or they display and manipulate the same image. One enhancement would be to implement a many sessions feature. For example, a possible approach for Co-Image is that the server records the session by the image name, and the users who display and manipulate the same image belong to the same session. So, a user may display and manipulate several images with the same/different people at same time.

- As it is now, Co-Draw and Co-Image provide the WYSIWIS interface. One interesting enhancement would be to provide customizable local views so that each user can tailor his/her display of a design without affecting the displays of the other participants. With this function, a participant can make changes locally first.

- As mentioned, the N-Talk service provides very good session management. A possible enhancement would be to implement a similar session management for

Co-Draw and Co-Image, individually. The other approach is to implement a single session management for all applications.

- Add more functions on the Co-Draw application. For example, support different color pens for each participant, add more shape items, add the scroll bar on the draw window.

- Another interesting enhancement would be the addition of audio/video media types, to provide more efficient communication between participants.

Multimedia collaboration systems try to utilize human senses to facilitate our communication with one another through computers. The goal is to achieve natural and expressive interaction—such as gesture, voice, gaze. The big performance challenge in these systems occurs when session participants continuously transmit video and voice streams, and the different media must be synchronized. Also, high bandwidth is required for the network. However, fiber-optic networks, coupled with improved computing and compression techniques, will soon be capable of delivering the audio/video media more quickly. With the development in networks, file servers, as well as increased processing power and operating systems, efficient multimedia communication will possible. Multimedia collaboration systems suggest a wide variety of potential applications in research, education, and business.

Bibliography

[1] S. R. Ahuja and J. R. Ensor. Coordination and control of multimedia conferencing. *IEEE Comm.*, 30(5):38–43, 1992.

[2] Vinod Anupam and Chandrajit L. Bajaj. A cscw infrastructure for scientific design and prototyping. Technical report, Purdue University, West Lafayette, USA, 1994.

[3] Vinod Anupam and Chandrajit L. Bajaj. Shastra: Multimedia collaborative design environment. *IEEE Multimedia*, 1(2):38–49, 1994.

[4] L. O. Barbora and N.D. Georganas. Multimedia services and applications. *European Trans. on Telecommunications*, 2(1):5–19, January 1991.

[5] Trudy E. Bell. Knowing what others are doing. *IEEE Spectrum*, 27(10):69–72, October 1990.

[6] Richard Bentley, Tom Rodden, Peter Sawyer, and Ian Sommerville. An architecture for tailoring cooperative multi-user displays. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work*, pages 187–194, November 1992.

[7] Kenneth P. Birman. The process group approach to reliable distributed computing. *Comm. of The ACM*, 36(11):37–53, December 1993.

[8] W. J. Clark. Multiport multimedia conferencing. *IEEE Comm.*, 30(5):44–50, May 1992.

[9] Terrence Crowley, Paul Milazzo, Ellie Baker, Harry Forsdick, and Raymond Tomlinson. Mmconf: An infrastructure for building shared multimedia applications. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work*, pages 329–342, October 1990.

[10] Peter Decker. *Managing in Turbulent Times*. Harper and Row, New York, 1992.

[11] Prasun Dewan and Rajiv Choudhary. A high-level and flexible framework for implementing multiuser user interface. *ACM Transaction on Information Systems*, 10(4):345–380, October 1992.

[12] A. Dix, J. Finley, G. Abowd, and R. Beale. *Human-Computer Interaction*. Prentice Hall, 1993.

[13] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of ACM Conference on Computer-Supported Cooperative Work*, pages 107–122, November 1992.

[14] C. Ellis, S. J. Gibbs, and G. Rein. Design and use of a group editor. In *Engineering for Human-Computer Interaction*, pages 13–25, September 1988.

[15] Clarence A. Ellis, Simon J. Gibbs, and Gail L. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1):38–58, January 1991.

[16] Robert S. Fish, Robert E. Karut, Mary D. P. Leland, and Michael Cohen. Quilt: a collaborative tool for cooperative writing. In *The Conference on Office Information Systems*, pages 30–37, March 1988.

[17] Jolene Galegher, Robert Kraut, and Carmen Egido. *Intellectual teamwork : social and technological foundations of cooperative work*. Hillsdale, N.J. : L. Erlbaum Associates, New York, 199o.

[18] G.O. Goodman and M.J. Abel. Collaboration research in scl. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, 1986.

[19] Saul Greenberg and David Marwood. Real time groupware as a distributed system: Concurrency control and its effect on the interface. Technical Report 94/534/03, University of Calgary, Calgary, Alberta, Canada, February 1994.

[20] Saul Greenberg, Mark Roseman, David Webster, and Ralph Bohnet. Issues and experiences designing and implementing two group drawing tools. In *Proceedings of Hawaii International Conference on System Sciences*, pages 138–150, January 1992.

[21] Hiroshi Ishii. Teamworkstation: Towards a seamless shared workspace. In *The Conference on Computer-Supported Cooperative Work*, pages 13–26, October 1990.

[22] Hiroshi Ishii and Naomi Miyake. Toward an open shared workspace: Computer and video fusion approach of teamworkstation. *Communications of the ACM*, 34(12):37–50, December 1991.

[23] R. Johansen. *Groupware*. Free Press, New York, 1988.

[24] Eric F. Johnson and Kevin Reichard. *Power Programming... Motif*. MIS Press, New York, 1993.

[25] W. E. Kackay and G. Davenport. Virtual video editing in interactive multimedia applications. *Communications of the ACM*, 32(7):802–810, July 1989.

[26] Michael Knister and Atul Prakash. Issues in the design of a toolkit for supporting multiple group editors. *The Usenix Association*, 6(2):135–166, 1993.

[27] Michael J. Knister and Atul Prakash. Distedit: A distributed toolkit for supporting multiple group editors. In *The Conference on Computer-Supported Cooperative Work*, pages 343–355, October 1990.

[28] Hannes P. Lubich. *Towards a CSCW Framework for Scientific Cooperation in Europe*. Springer-Verlag, Heidelberg, 1995.

[29] Christine M. Neuwirth, David S. Kaufer, Ravinder Chandhok, and James H. Morris. Issues in the design of computer support for co-authoring and commenting. In *The Conference on Computer-Supported Cooperative Work*, pages 183–195, October 1990.

[30] R. E. Newman-Wolfe and H. K. Pelimuhandiram. Mace: A fine-grained concurrent editor. In *Proceedings of the ACM/IEEE Conference on Computer-Supported Cooperative Work*, pages 240–254, November 1991.

[31] Atul Prakash. Undoing actions in collaborative work: Framework and experience. Technical Report CSE-TR-196-94, University of Michigan, Ann Arbor, MI 48109-2122, 1994.

[32] Hill Ralph D. The abstraction-link-view paradigm: Using constraints to connect user interfaces to applications. In *CHI'92 ACM Conference*, pages 335–342, May 1992.

[33] Mark Roseman and Saul Greenberg. Groupkit: A groupware toolkit for building real-time conferencing applications. In *Proceeding of the Conference on Computer-Supported Cooperative Work*, pages 43–50, October 1992.

[34] Mark Roseman and Saul Greenberg. Registration for real-time groupware. Department of Computer Science, University of Calgary, 1992.

[35] Stephen A.R. Scrivener. *Computer-Supported Cooperative Work*. Avebury Technical, England, 1994.

[36] Ian Sommerville, Richard Bentley, Tom Rodden, and Pete Sawyer. Architectural support for cooperative multi-user interfaces. Technical Report CSCW/11/93, Lancaster University, Lancaster, UK, 1993.

[37] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar. Wysiwis revised: Early experiences with multiuser interfaces. *ACM Trans. on Office Information Systems*, 5(2):147–167, April 1987.

[38] Mark Stefik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30(1):32–47, July 1987.

[39] John C. Tang and Scott L. Minneman. Videodraw: A video interface for collaborative drawing. In *Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing(CHI'90)*, pages 313–320, 1990.

[40] John C. Tang and Scott L. Minneman. Videowhite: Video shadows to support remote collaboration. In *Proceedings of the ACM/SIGCHI Conference on Human Factors in Computing(CHI'91)*, pages 315–322, 1991.

[41] Tore Urnes and Roy Nejabi. Tools for implementing groupware: Survey and evalation. Technical report, York University, North York, Ontario, Canada, 1994.

[42] Kazuo Watabe, Shiro Salata, Kazutoshi Maeno, Hideyuki Fukuoka, and Toyoko Ohmori. Distributed multiparty desktop conferencing system: Mermaid. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 27–38, October 1990.

[43] P. A. Wilson and CCTA Advanced Concepts Branch. *Computer Supported Cooperative Work: An Introduction.* Intellect Publisher, Oxford, UK, 1991.